RasPi

DESIGN
BUILD
CODE

4

Get hands-on with your Raspberry Pi

Code your own SPACE INVADERS

Plus
5 Amazing add-ons

60 PAGES OF PI

Welcome

They say the only way to really learn how something works is to take it all apart, have a look at the components and then put it back together. Well that's the approach we're taking this month with our special guide to programming one of the most popular arcade games ever made. In this issue of **RasPi** we're showing you how to code your own Space Invaders clone piece by piece, walking you through all of the main code blocks and explaining what they do. By the end, you'll have mastered the fundamentals of Python games in just 300 lines of code. We also have some amazing Raspberry Pi modules to show you, plus guides to setting up wireless access points and more. Enjoy your new issue!

Gavin Thomas

Deputy Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

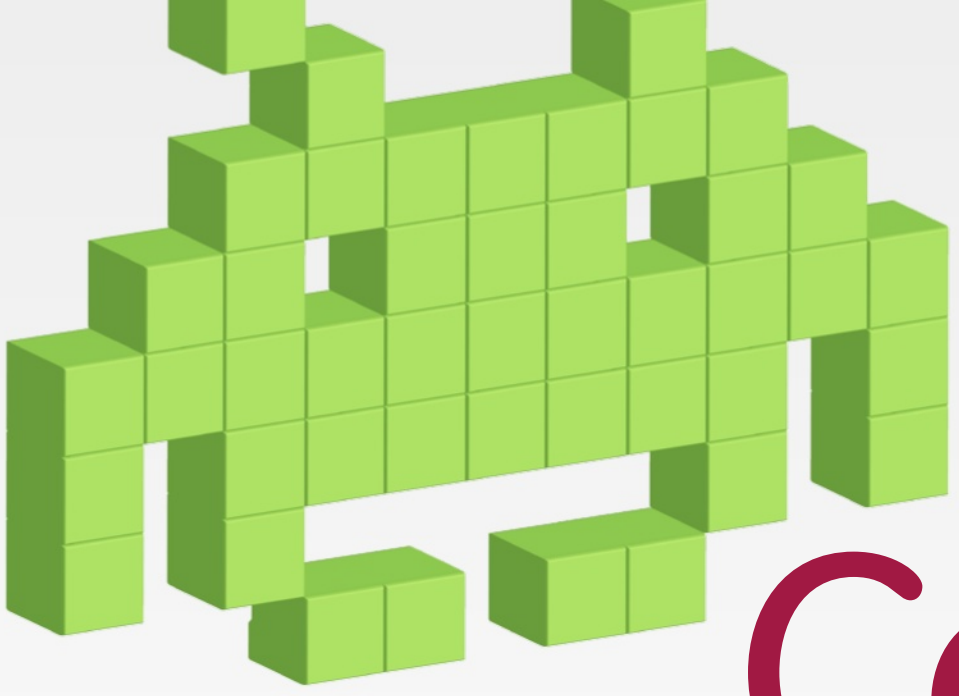
Learn to make and code gadgets with Raspberry Pi

From the makers of

LinuxUser & Developer

Join the conversation at...

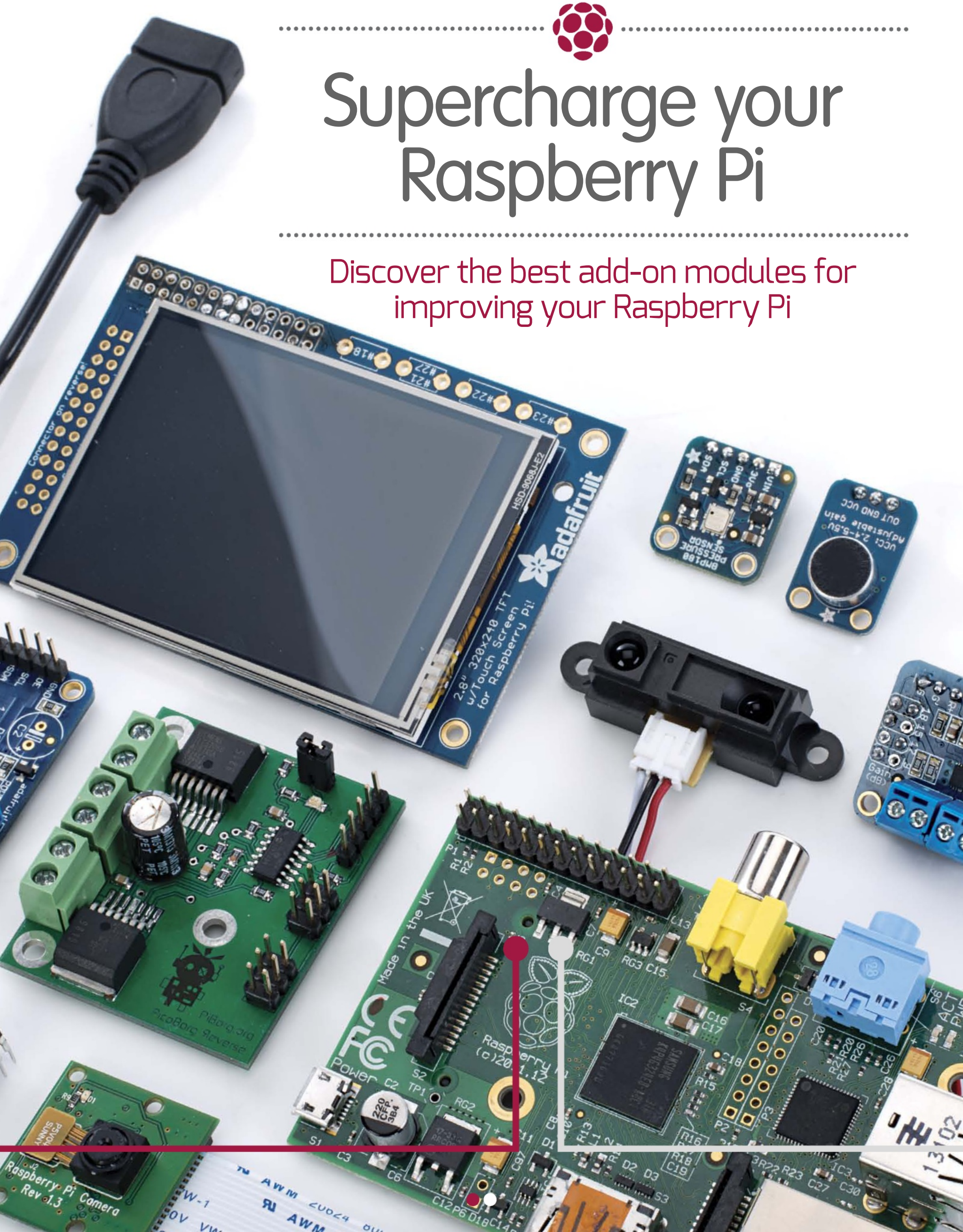
@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk

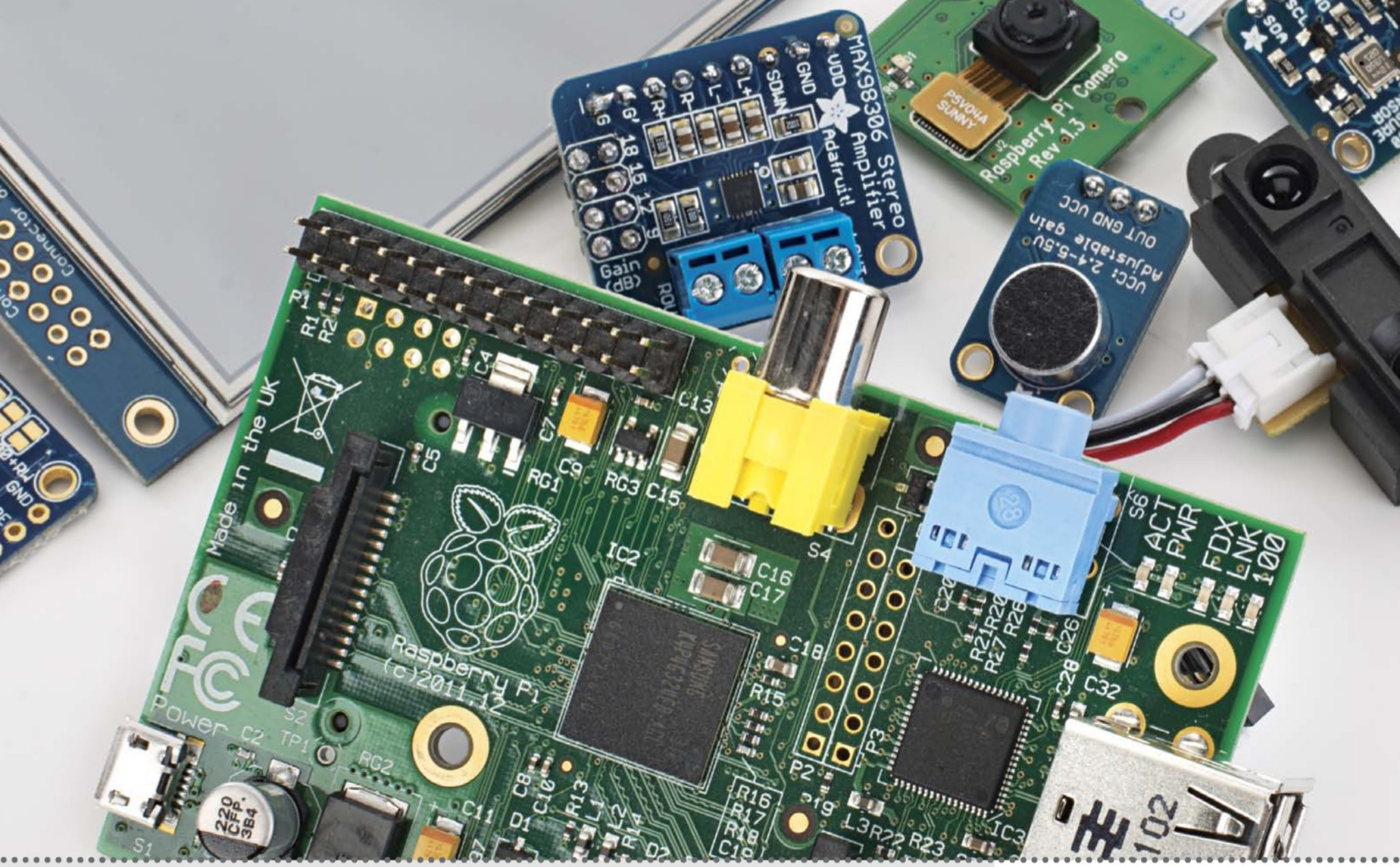


Contents

Supercharge your Raspberry Pi

Discover the best add-on modules for improving your Raspberry Pi





The Raspberry Pi is a fantastic piece of British engineering. Just five years ago the concept of a credit card-sized computer capable of playing video at 1080p and offering a full desktop PC experience would have been considered witchcraft. Claiming that it was able to do all this and more at a retail price of a little under £25 would have seen you committed.

It's clear that the Raspberry Pi has been a revolution for small form factor computing, but it's not quite perfect – only so much technology can be packed into a tiny space, after all. For enthusiasts and makers there's a growing list of extra-curricular add-ons that can help make their projects, gadgets and Intenet Of Things devices come to life. We've selected five of the most essential and over the next few pages we'll show you where to get them and how to get started…

> "It's clear that the Raspberry Pi has been a revolution for small form factor computing"

Read analog inputs

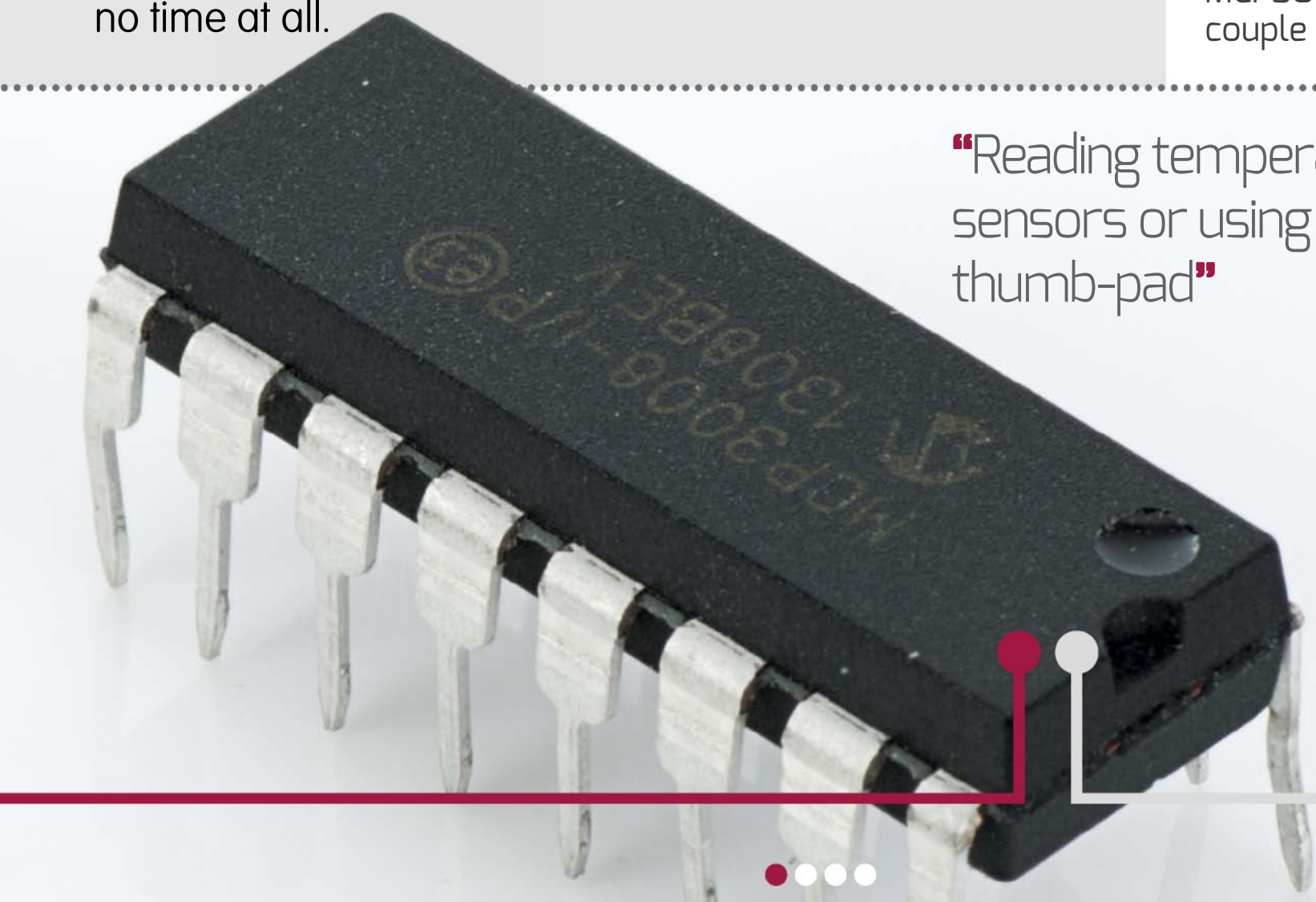
Unlike other microcontrollers, the Pi can't read analog sensors – but adding this functionality is really affordable

What is it?

If you want to add the ability to read analog sensors for your Raspberry Pi projects you'll need a very particular piece of hardware – an analog to digital converter (ADC). The cheapest and simplest way to implement this with the Raspberry Pi is using an ADC chip like the MCP3008, which adds eight analog inputs at the cost of just four GPIO pins. With a small selection of prototyping parts and a sprinkling of Python code you could be reading temperature sensors or using an analog thumb-pad to create your own games controller in next to no time at all.

THE PROJECT ESSENTIALS

MCP3008
www.cpc.co.uk
Breadboard
Prototyping cables (male to female)



Below Adafruit sells MCP3008 for just a couple of pounds

> "Reading temperature sensors or using an analog thumb-pad"

Why do it?

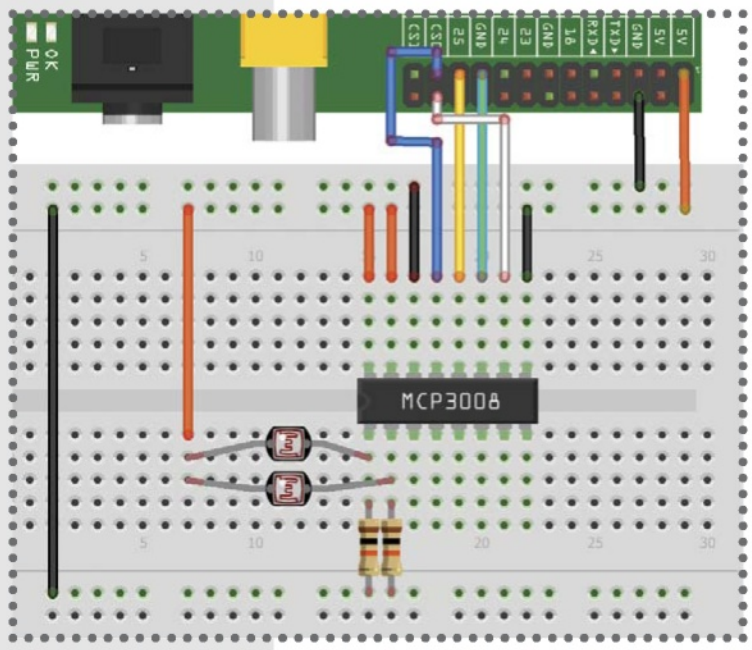
The Raspberry Pi's GPIO pins are great – they allow you to interact with the physical world in really cool ways, simply by turning the voltage to its various pins on and off. With its single pulse-width-modulation-capable (PWM) pin, the Raspberry Pi can even offer relatively fine-grain control to change the brightness of the LED light or alter the speed of a DC motor – but for fans of electronics-based projects, analog sensing is something that's sorely missed and readily available on Arduino and similar microcontrollers. With it you can get very precise feedback from a potentiometer (a twisty knob like a volume control) or get accurate readings from something like an IR reflectance sensor to help your robot not bump into things. You could also use an ADC-enabled Raspberry Pi to check the ambient light to turn a night-light on automatically using a cheap light dependent resistor (LDR), otherwise known as photocells, as we've demonstrated on our example breadboard.

"You can get very precise feedback from a potentiometer or get accurate readings from an IR reflectance sensor"

How to use it

The MCP3008 ADC does take a little bit of setting up, but once it's set up it's very easy to start using. This tiny chip can be bought for as little as £2 from most good electronics stockists. Since it's unlikely that you will want to hardwire it into your initial projects, we recommend using a breadboard and male-to-female prototyping cables to put it all together as we've demonstrated in our Fritzing diagram. Although the diagram looks relatively complex, we're actually only using four GPIO pins to control the chip logic and the 3.3V and Ground pins to co-ordinate power to the other four pins along the right side of the chip (the small semi-circle at one end of the chip denotes its 'top'). The pins down the entire left side of the chip are for your analog inputs.

Below Connect your chip to the GPIO as shown. Besides power, there are only 4 connections



Install the necessary software

01 Clear the blacklist

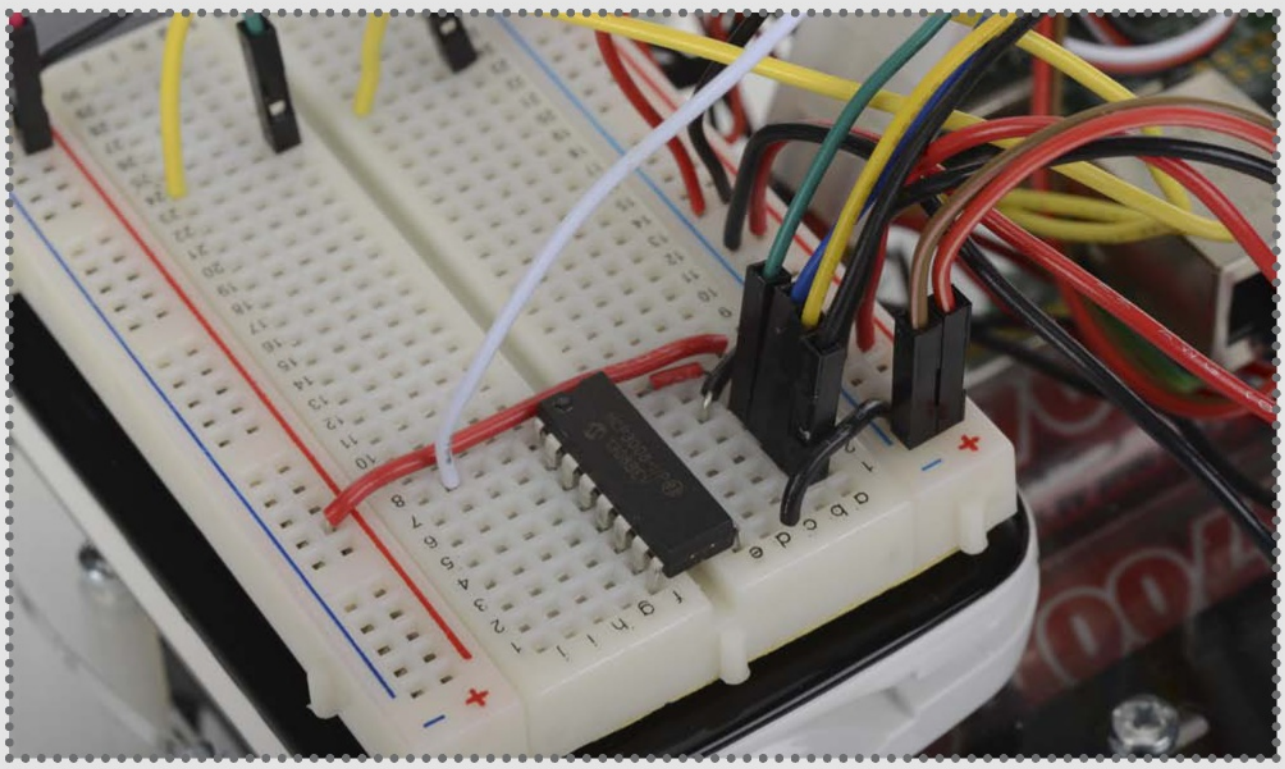
With a terminal window open, type: `sudo nano /etc/modprobe.d/raspi-blacklist.conf`. Add a hash symbol (#) to the start of each line in the file and then press CTRL+X, then Y and press Enter to save and exit the file.

02 Install Python dev tools and spidev

There are a couple of critical packages we need. At the terminal type `sudo apt-get install python-dev python-pip`. Once the packages are installed type `sudo pip install spidev` then finally `sudo reboot` to restart your Raspberry Pi.

03 Read the sensors

To print the result from any analog sensor connected to the left side of the MCP3008, simply use the following code example – write it in Leafpad and save it as adc_test.py. In the script we are reading the first two pins to see the results from our two photocells. Run it with `sudo python adc_test.py` to test it.



Left As you can see, the MCP3008 takes up very little room on a small breadboard

"The pins down the entire left side of the chip are for your analog inputs"

The Code READ THE SENSORS

```
import spidev
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
spi = spidev.SpiDev()
spi.open(0,0)

def readadc(adcnum):
    if ((adcnum > 7) or (adcnum < 0)):
        return -1
    r = spi.xfer2([1,(8+adcnum)<<4,0])
    adcout = ((r[1]&3) << 8) + r[2]
    return adcout

while True:
    print "Photo Cell 1: " + str(readadc(0))
    print "Photo Cell 2: " + str(readadc(1))
    time.sleep(1)
```

"In the script we are reading the first two pins to see the results from our two photocells"

Use an analog distance sensor

Use IR reflectance sensors to detect surroundings

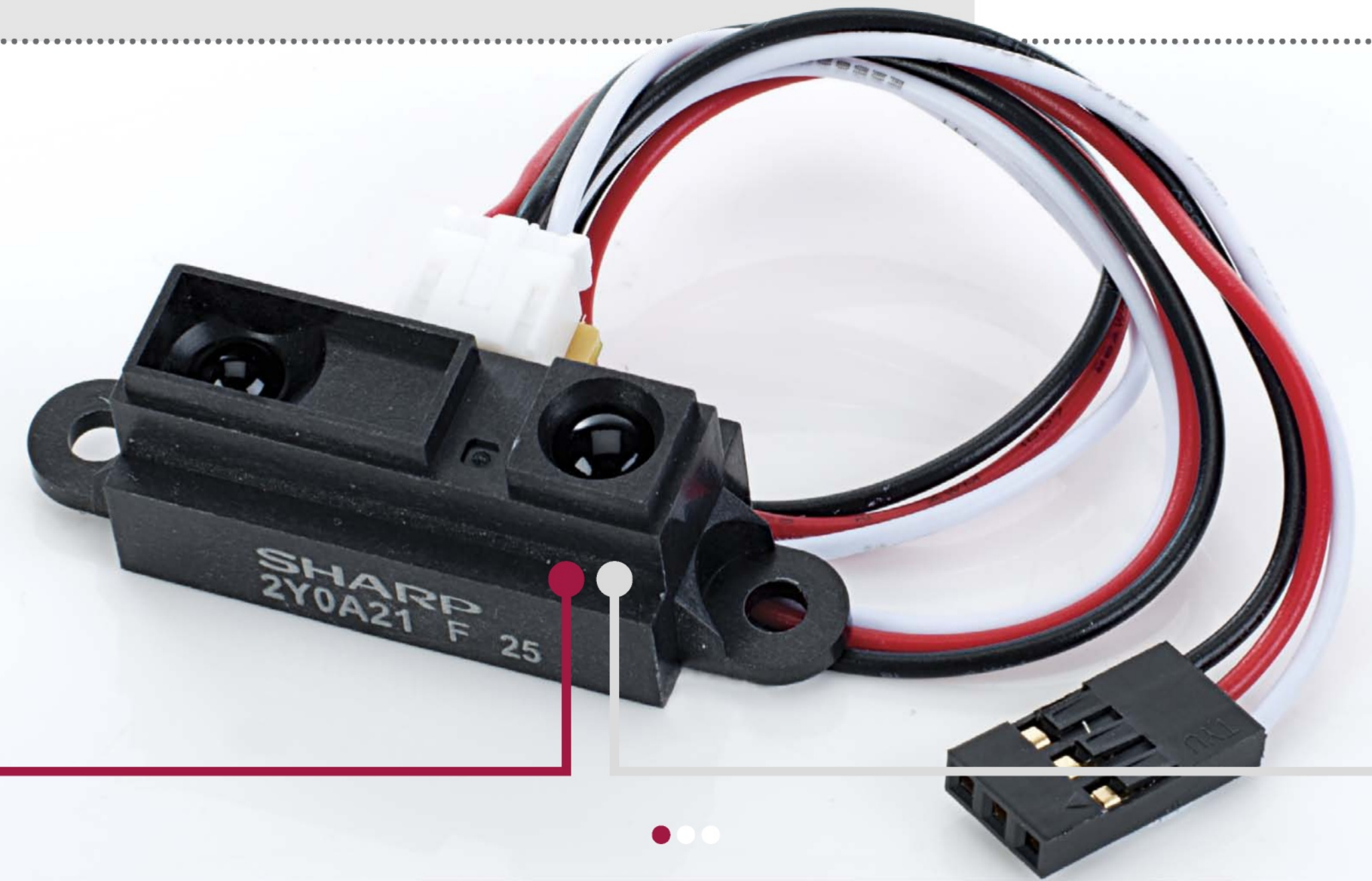
What is it?

An IR reflectance sensor such as the Sharp GP2D120 offers a great way to sense the world, using the infra-red light spectrum to 'ping' its surroundings in much the same way as a bat employs echolocation to avoid obstacles in the dark. The closer an obstacle is to the sensor, the higher the reading you'd receive from its analog reading (between 0 and 1023).

THE PROJECT ESSENTIALS

Sharp GP2D120 IR Range Finder
www.robotbits.co.uk

Below The GP2D120 has an effective range of 4-30cm and draws 33mA of current



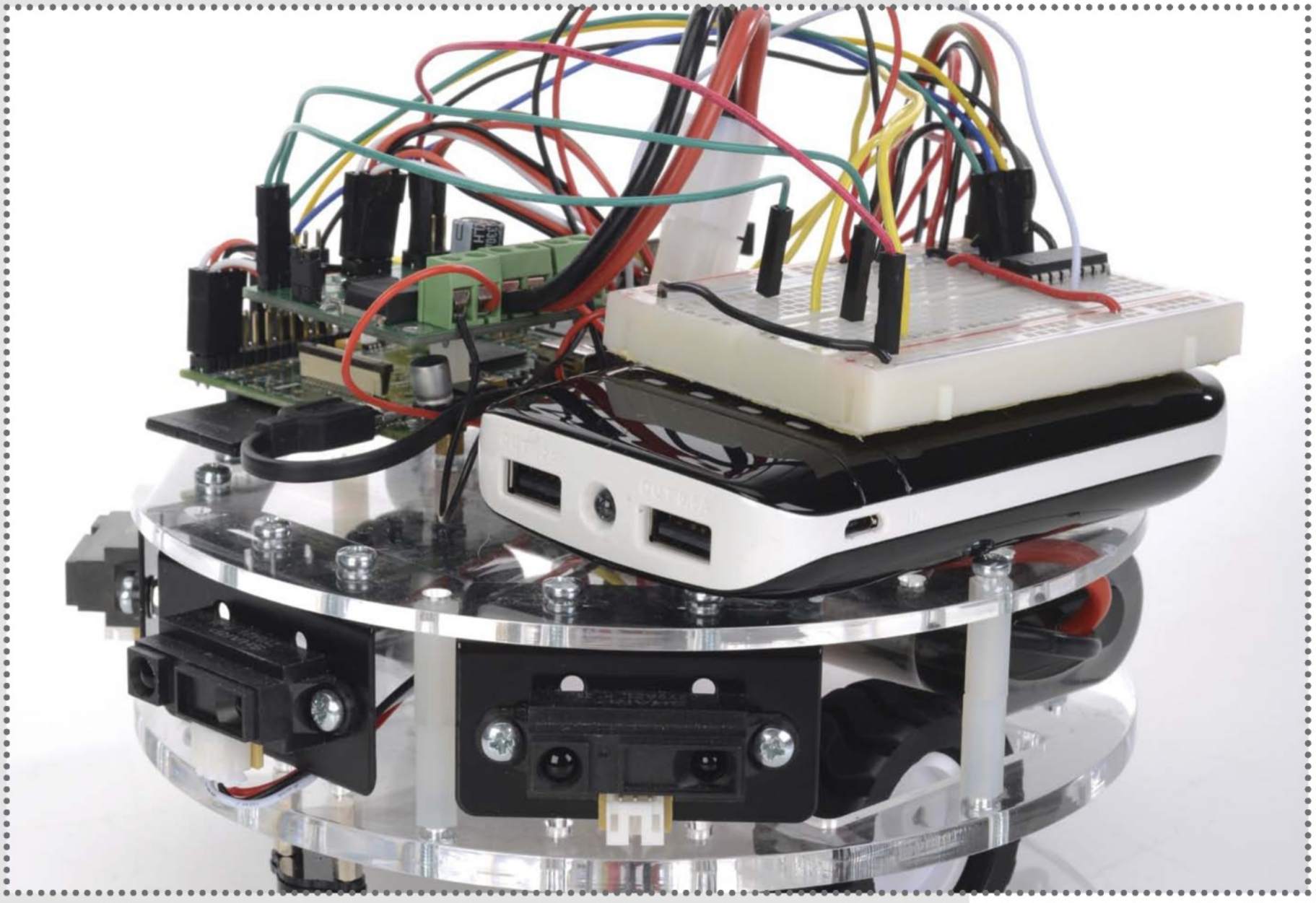
Why do it?

These kinds of sensors are perfect for Raspberry Pi projects that require any kind of acknowledgement of its surroundings. For example, you could place a sensor near a bird perch and have it trigger the camera module to take a picture should a reading indicate activity in your target area. Alternatively, you could employ several sensors on an autonomous robot – if the left sensor gets a high reading, it's trivial to then tell your robot to turn right.

How to use it

We're going to use the robot example and the test code we produced for the MCP3008 to show how easy it is to read and decipher the readings from these sensors to make a robot make seemingly intelligent decisions.

"Sense the world using the infra-red light spectrum to 'ping' surroundings"



Below Analog IR Ranger Finders like these are perfect for making an autonomous bot

The Code USE AN ANALOG DISTANCE SENSOR

```
import spidev
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
spi = spidev.SpiDev()
spi.open(0,0)

front = 0
trigger = 450

def readadc(adcnum):
    if ((adcnum > 7) or (adcnum < 0)):
        return -1
    r = spi.xfer2([1,(8+adcnum)<<4,0])
    adcout = ((r[1]&3) << 8) + r[2]
    return adcout

def read_sensor():
    result = readadc(front)
    return result

while True:
    reading1 = read_sensor()
        if reading1 > trigger:
            print "Sensor", str(reading1), "triggered: Run!"
        else:
            print "Nothing to see here."
        time.sleep(0.5)
```

"You could place a sensor near a bird perch and have it trigger the camera module to take a picture"

Take pictures with your Pi

Add an eye to your Pi and open it up to photos, videos and optical sensing

What is it?

Released in 2013, the Raspberry Pi camera board is a small printed circuit board (PCB) which houses a tiny digital camera sensor of 1.3 megapixels. It's able to take photos, video and display a live preview of what the camera can see.

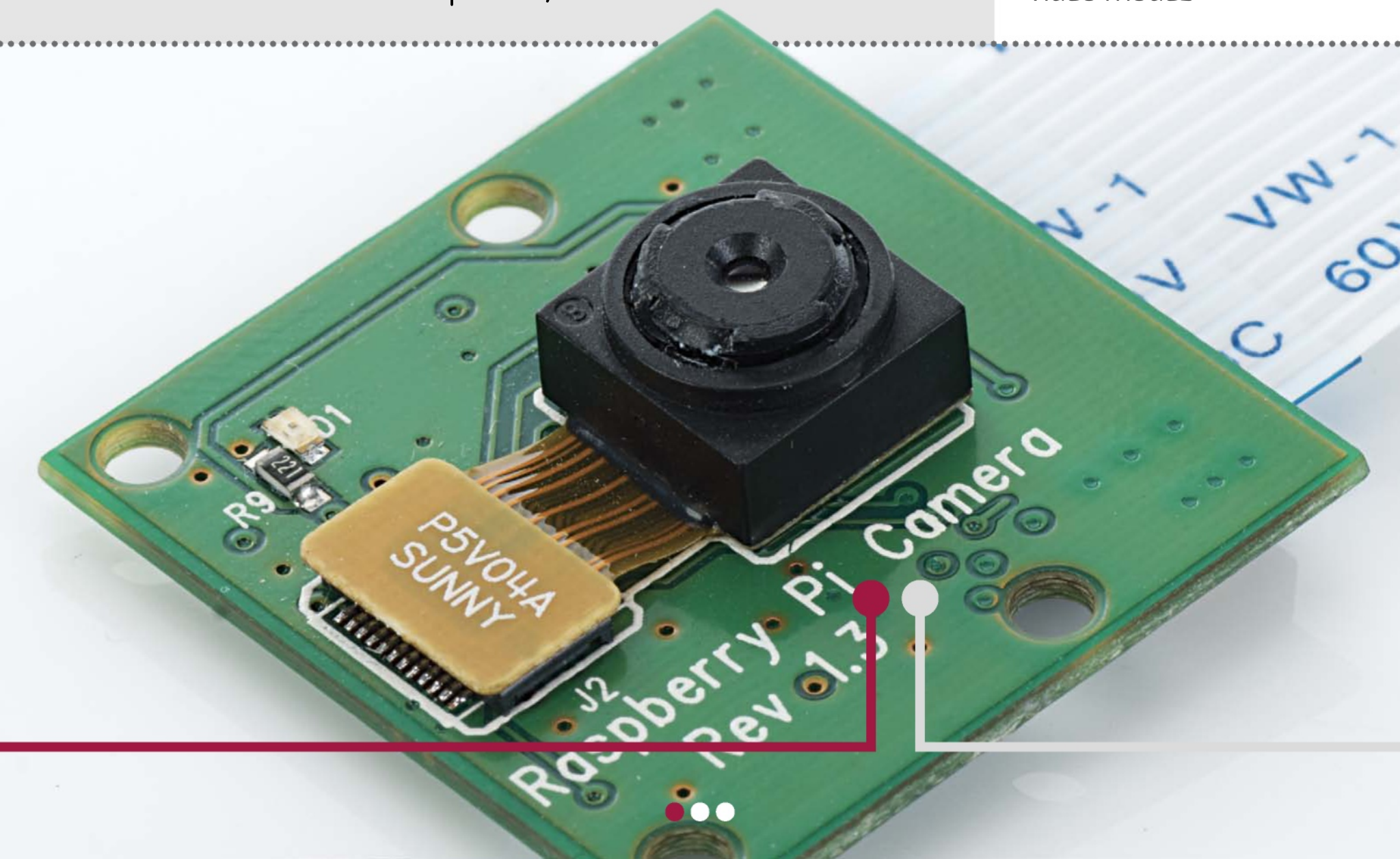
Why do it?

Because of the size and portability of the Pi, it's almost as convenient as a camera phone, albeit with more controls

THE PROJECT ESSENTIALS

Latest version of Raspbian
www.raspbian.org

Raspberry Pi Camera module
www.cpc.farnell.com



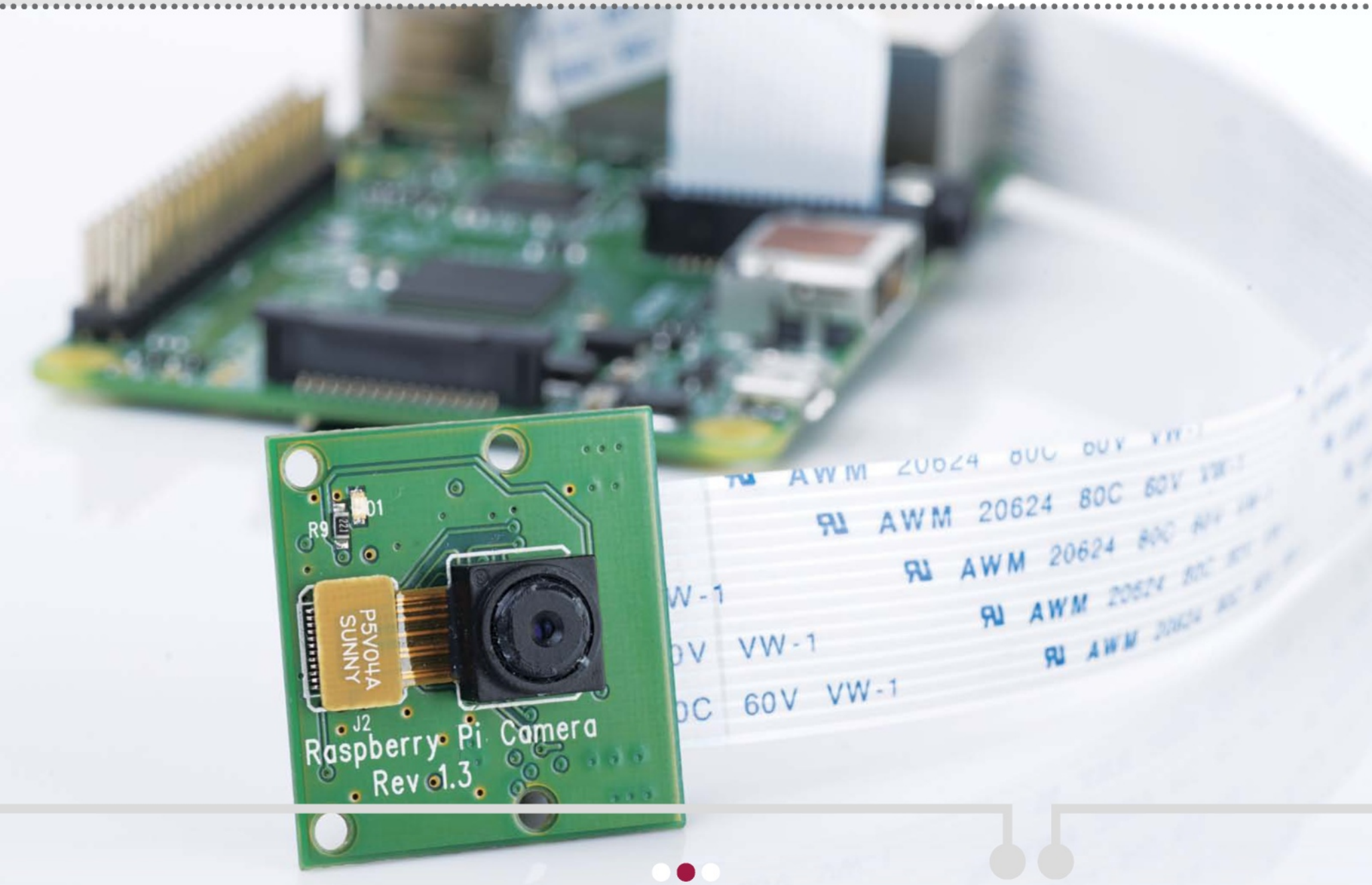
Below The camera supports 1080p30, 720p60 and VGA90 video modes

via the specialised libraries and software. Taking pictures and video could be done with a webcam, however that would have to take up a USB port. The camera board slots into a specific video-in connector that doesn't take up any of the conventional ports, a connector specifically designed with the then future camera in mind.

How to use it

Turn off your Raspberry Pi and make sure you disconnect the power. In between the HDMI port and the ethernet port there's a special connector. Pull up the tabs on both sides to open up the port and slot the ribbon into the opening with the silver connectors facing towards the HDMI. Gently push down the tabs to secure it and then connect the Raspberry Pi back up to power. To start using it, you will need to activate the camera module within the config settings.

> "The camera board slots into a specific video-in connector that doesn't take up any of the conventional ports"



Below Check the camera's docs to find commands for effects: **bit.ly/1vDF1Vl**

Enable the camera module

01 Update your Pi

You will need to make sure everything is up to date. Open the LXTerminal and update the firmware followed by the software:

```
$ sudo rpi-update
$ sudo apt-get update && sudo apt-get upgrade
```

02 Enter configuration

Once the updates are complete you can update the configuration files. This is the same one you saw after installing Raspbian. Staying in the terminal, open this with:

```
$ raspi-config
```

03 Enable the camera module

Go to the Enable Camera option and change the setting to Enable. Then, go to Finish on the main interface and it will reboot your Pi with the camera working after login. The Pi camera is now accessible via the command line and its operation is simple. To take a picture, use:

```
$ raspistill -o [imagename].png
```

This will display a five second preview before saving an image. Videos can be shot with:

```
$ raspivid -o [videoname].h264
```

…which records for five seconds. There are options you can add to each command that let you control exposure, preview length, framerate and resolution. There's also the picamera Python module that enables you to control it via Python.

"To start using it, you will need to activate the camera module within the config settings"

Add a power switch

Turn the Pi on and off properly with no fuss

What is it?

It's simply a binary switch sitting between a male and female USB A port: the connector present on every USB cable. In our case, the A port connects to source of the power for the Pi, allowing it to interrupt the flow of electricity and effectively turn it off and on.

Why do it?

As many users of the Raspberry Pi already know, once the Pi is turned off you will need to disconnect and reconnect the power in order for it to start booting up again. This switch allows you to skip this task and do it in a more traditional manner.

How to use it?

The main component in this setup is the USB Cable with Switch from Pimoroni, however you'll also need a micro USB cable to connect to the female end and a mains adapter that allows you to directly plug a USB A port into it.

THE PROJECT ESSENTIALS

USB Cable with Switch
www.shop.pimoroni.com

Below Illuminated LED power buttons are also available

Control servos

With just one PWM-capable pin, the Pi's servo support is limited at best

What is it?

Adafruit's 16-channel Servo/PWM Driver board is an impressive piece of kit. With it you can control up to 16 hobby servos using just two pins on your Pi (via I2C). You can daisy-chain 62 of these boards together on a single Pi, so you are able to control a staggering 992 PWM servos or LED lights at the same time.

Why do it?

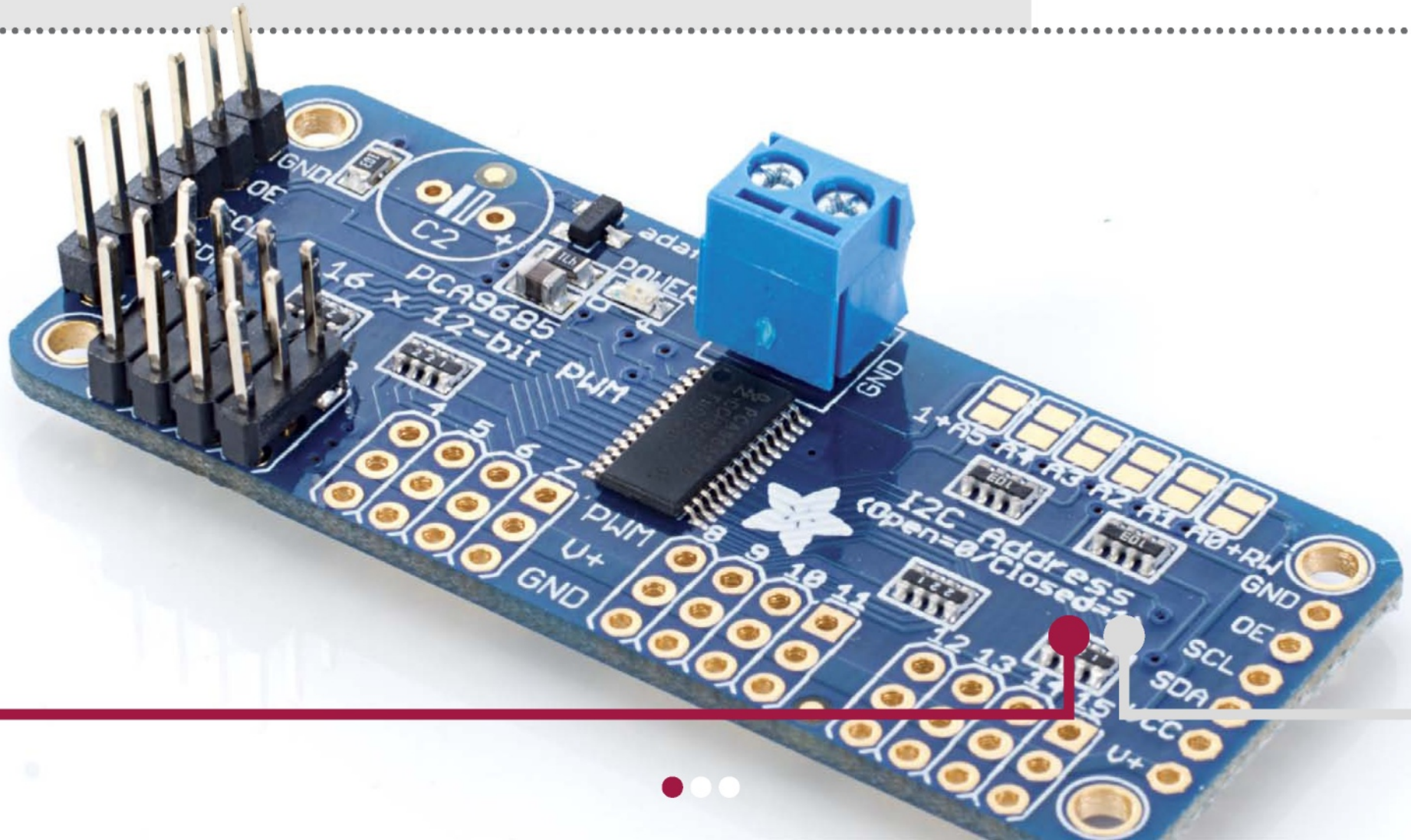
PWM (Pulse Width Modulation) is a challenge for the Pi as it only has a single compatible pin. Since the Pi isn't a real-time device like an Arduino, software-driven PWM

THE PROJECT ESSENTIALS

Adafruit 16-channel PWM/Servo Driver board
www.shop.pimoroni.com

Hobby servos
www.modmypi.com

Battery holder
www.shop.pimoroni.com



Below These come in handy for complex robotics projects

can be hit and miss. While it works well controlling LEDs, controlling a servo on a general-purpose computer like the Pi can lead to undesired results like jittering.

How to use it

If you want to be able to make anything that requires precise moving parts, a dedicated hardware solution to PWM is a must, and that's exactly what Adafruit's 16-channel Servo/PWM Driver board offers. Once all of the soldering is done, you need to set up your Pi to talk to the board with I2C. Check out our step-by-step for software usage on the next page.

How it works

With the software installed and your driver board hooked up with a battery pack attached you can experiment with servos in your projects.

With the third-party Python library we install in the following steps, it's easy to program the servos. See **bit.ly/1nNDaXC** for more details.

Below Rapiro uses a total of 12 servos and a servo motor controller board

Grab the Python library

01 Install dependencies

To set up I2C you need to make sure I2C communication isn't blacklisted. Type `sudo nano /etc/modprobe.d/raspi-blacklist.conf` and comment (add an #) to the start of the line 'blacklist i2c-bcm2708'.

Next you need to install SMBus and I2C tools. You can do this in the terminal with the following command: `sudo apt-get install python-smbus i2c-tools`

02 Grab the Python library

Since the Raspbian distro comes complete with git, it's really easy to grab the official Adafruit Python library with the following command in the terminal:

```
git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

You can navigate into the folder with:

```
cd Adafruit-Raspberry-Pi-Python-Code/Adafruit_PWM_Servo_Driver
```

03 Install a third-party library

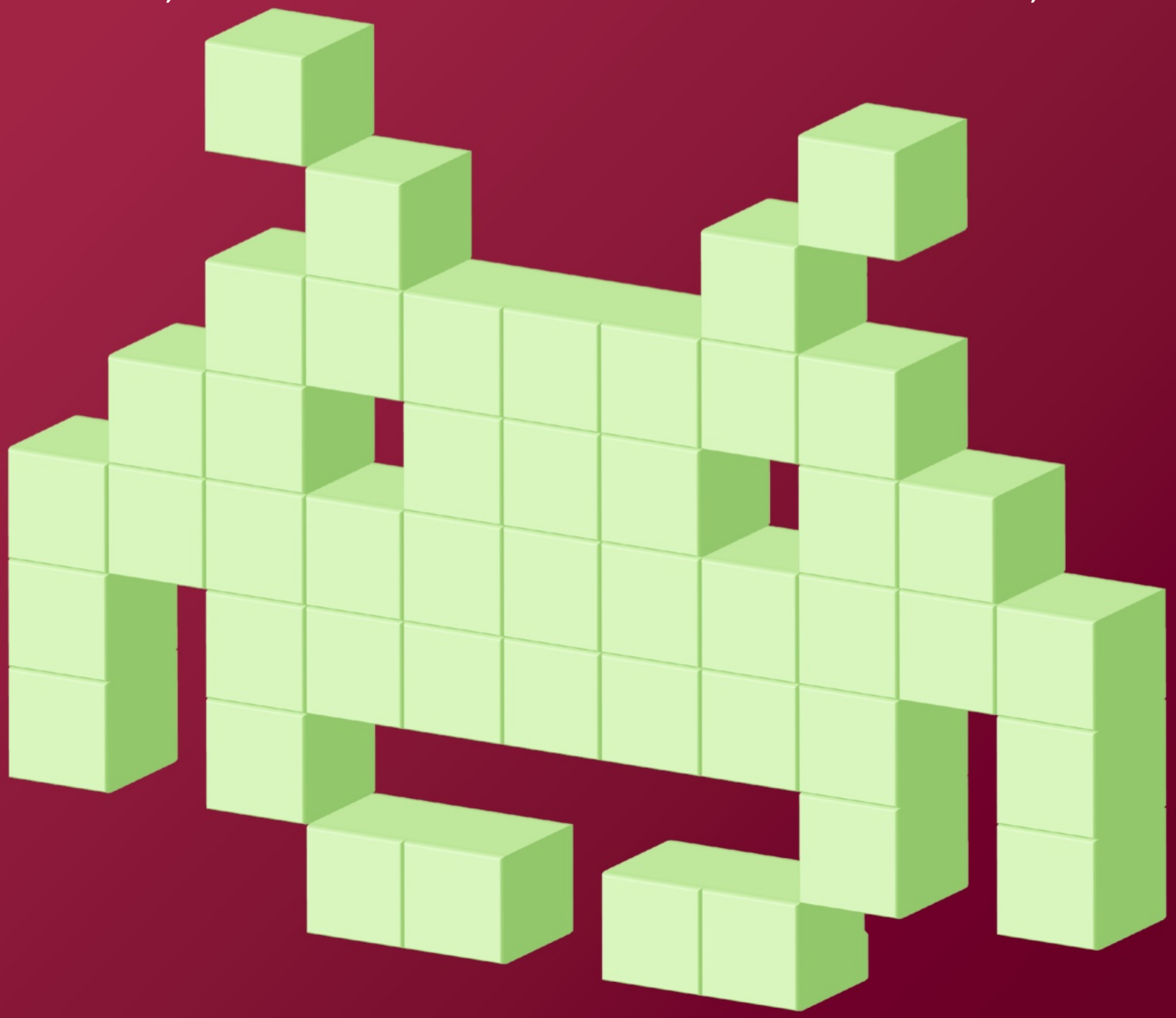
While the official library is good, it's incredibly basic and doesn't make servo operation easy. Luckily, there's a small third-party library that uses the important bits from the Adafruit library and builds on it. Grab it using git from the terminal window with:

```
git clone https://github.com/labatrockwell/raspberrypi-experiments.git
```

"Once all of the soldering is done, you need to set up your Pi to talk to the board with I2C"

Program a Space Invaders clone

Write your own RasPi shooter in 300 lines of Python



"Experimenting with a familiar and relatively simply project is a very useful exercise"

When you're learning to program in a new language or trying to master a new module, experimenting with a familiar and relatively simply project is a very useful exercise to help expand your understanding of the tools you're using. Our Space Invaders clone is one such example that lends itself perfectly to Python and the Pygame module – it's a simple game with almost universally understood rules and logic. While the Invaders make their way down the screen towards you, it's your job to pick them off while dodging their random fire. When one wave is conquered, another faster, more aggressive wave appears. We've tried to use many features of Pygame, which is designed to make the creation of games and interactive apps easier. We've extensively used the Sprite class, which saves dozens of lines of extra code by making collision detection simple and updating the screen (and its many actors) a single-line command. Next time we'll be adding animation and sound effects to our game…

01 Setting up dependencies

If you're looking to get a better understanding of programming games with Python and Pygame, we strongly recommend you copy the Pivaders code in this tutorial into your own program. It's great practice and gives you a chance to tweak elements of the game to suit you, be it a different ship image, changing the difficulty or the ways the alien waves behave. If you just want to play the game, that's easily achieved too, though. Either way, the game's only dependency is Pygame, which (if it isn't already) can be installed from the terminal by typing:

```
sudo apt-get install python-pygame
```

THE PROJECT ESSENTIALS

Latest Raspbian Image
raspberrypi.org/downloads

Python
python.org/doc

Pygame
pygame.org/docs

Skills to learn

Python
You'll need a basic grounding in Python to follow along with the code. Try **codecademy.com**

Classes
Hopefully the article shows how useful classes can be to create and manipulate multiple objects with ease.

02 Downloading the project

For Pivaders we've used Git, a brilliant form of version control used to safely store the game files and retain historical versions of your code. Git should already be installed on your Pi; if not, you can acquire it by typing:

```
sudo apt-get install git
```

As well as act as caretaker for your code, Git enables you to clone other people's projects so you can work on them. To clone Pivaders, go to your home folder in the terminal (cd ~), make a directory for the project (mkdir pivaders), enter it (cd pivaders) and type:

```
git pull https://github.com/LinuxUserMag/pivaders.git
```

03 Testing Pivaders

With Pygame installed and the project cloned to your machine (we've also hosted the .zip on our website – just head over to **bit.ly/11k5f2x** to start downloadling the files), you can take it for a quick test drive to make sure everything's set up properly. All you need to do is type python pivaders.py from within the pivaders directory in the terminal to get started. You can start the game with the space bar, shoot with the same button and simply use the left and right arrows on your keyboard to move your ship left and right.

04 Creating your own clone

Once you've racked up a good high score (anything over 2,000 points is respectable) and got to know our simple implementation, you'll get more from following along with and exploring the code and our brief explanations of what's going on. For those who want to make their own project, create a new project folder and use either IDLE or Leafpad (or perhaps install Geany) to create and save a .py file of your own.

"We've extensively used the Sprite class, which saves dozens of lines of extra code by making collision detection simple and updating the screen a single-line command"

05 Global variables & tuples

Once we've imported the modules we need for the project, there's quite a long list of variables in block capitals. The capitals denote that these variables are constants (or global variables). These are important numbers that never change – they represent things referred to regularly in the code, like colours, block sizes and resolution. You'll also notice that colours and sizes hold multiple numbers in braces – these are tuples. You could use square brackets (to make them lists), but we use tuples here since they're immutable, which means you can't reassign individual items within them. Perfect for constants, which aren't designed to change anyway.

06 Classes – part 1

A class is essentially a blueprint for an object you'd like to make. In the case of our player, it contains all the required info, from which you can make multiple copies (we create a player instance in the make_player() method halfway through the project). The great thing about the classes in Pivaders is that they inherit lots of capabilities and shortcuts from Pygame's Sprite class, as denoted by the pygame.sprite.Sprite found within the braces of the first line of the class. You can read the docs to learn more about the Sprite class via **www.pygame.org/docs/ref/sprite.html.**

07 Classes – part 2

In Pivader's classes, besides creating the required attributes – these are simply variables in classes – for the object (be it a player, an alien, some ammo or a block), you'll also notice all the classes have an update() method apart from the Block class (a method is a function within a class). The update() method is called in

"A class is essentially a blueprint for an object you'd like to make. In the case of our player, it contains all the required info, from which you can make multiple copies"

every loop through the main game (we've called ours main_loop()) and simply asks the iteration of the class we've created to move. In the case of a bullet from the Ammo class, we're asking it to move down the screen. If it goes off either the top or bottom of the screen, we destroy it (since we don't need it any more).

08 Ammo

What's most interesting about classes, though, is that you can use one class to create lots of different things. You could, for example, have a pet class. From that class you could create a cat (that meows) and a dog (that barks). They're different in many ways, but they're both furry and have four legs, so can be created from the same parent class. We've done exactly that with our Ammo class, using it to create both the player bullets and the alien missiles. They're different colours and they shoot in opposite directions, but they're fundamentally one and the same. This saves us creating extra unnecessary code and ensures consistent behaviour between objects we create.

"We've used our Ammo class to create both the player bullets and the alien missiles"

Below We used widely available open source art and fonts to make the game

09 The game

Our final class is called Game. This is where all the main functionality of the game itself comes in, but remember, so far this is still just a list of ingredients – nothing can actually happen until a 'Game' object is created (right at the bottom of the code). The Game class is where the central mass of the game resides, so we initialise Pygame, set the imagery for our protagonist and extraterrestrial antagonist and create some GameState attributes that we use to control key aspects of external classes, like changing the player's vector (direction) and deciding if we need to return to the start screen, among other things.

10 The main loop

There are a lot of methods (class functions) in the Game class, and each is designed to control a particular aspect of either setting up the game or the gameplay itself. The actual logic that dictates what happens within any one round of the game is actually contained in the main_loop() method right at the bottom of the pivaders.py script and is the key to unlocking exactly what variables and functions you need for your game. Starting at the top of main_loop() and working line-by-line down to its last line, you can see exactly what's being evaluated 20 times every second when you're playing the game.

11 Main loop key logic – part 1

Firstly the game checks that the end_game attribute is false – if it's true, the entire loop in main_loop() is skipped and we go straight to pygame.quit(), exiting the game. This flag is set to true only if the player closes the game window or presses the Esc key when on the start_screen. Assuming end_game and

"The logic that dictates what happens within any one round of the game is actually contained in the main_loop() method right at the bottom of the pivaders.py script"

start_screen are false, the main loop can start proper, with the control() method, which checks to see if the location of the player needs to change. Next we attempt to make an enemy missile and we use the random module to limit the number of missiles that can be created. Next we call the update() method for each and every actor on the screen using a simple for loop. This makes sure everyone's up to date and moved before we check collisions in calc_collisions().

"We call the update() method for each and every actor on the screen using a simple for loop"

12 Main loop key logic – part 2

Once collisions have been calculated, we need to see if the game is still meant to continue. We do so with is_dead() and defenses_breached() – if either of these methods returns true, we know we need to return to the start screen. On the other hand, we also need to check to see if we've killed all the aliens, from within win_round(). Assuming we're not dead, but the aliens are, we know we can call the next_round() method, which creates a fresh batch of aliens and increases their speed around the screen. Finally, we refresh the screen so everything that's been moved, shot or killed can be updated or removed from the screen. Remember, the main loop happens 20 times a second – so the fact we don't call for the screen to update right at the end of the loop is of no consequence.

The Code CODING PIVADERS

```
#!/usr/bin/env python2
import pygame, random
BLACK = (0, 0, 0)
BLUE = (0, 0, 255)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
ALIEN_SIZE = (30, 40)
ALIEN_SPACER = 20
BARRIER_ROW = 10
BARRIER_COLUMN = 4
BULLET_SIZE  = (5, 10)
MISSILE_SIZE = (5, 5)
BLOCK_SIZE = (10, 10)
RES = (800, 600)
```

```
class Player(pygame.sprite.Sprite):
  def __init__(self):
    pygame.sprite.Sprite.__init__(self)
    self.size = (60, 55)
    self.rect = self.image.get_rect()
    self.rect.x = (RES[0] / 2) - (self.size[0] / 2)
    self.rect.y = 520
    self.travel = 7
    self.speed = 350
    self.time = pygame.time.get_ticks()
  def update(self):
    self.rect.x += GameState.vector * self.travel
    if self.rect.x < 0:
      self.rect.x = 0
    elif self.rect.x > RES[0] - self.size[0]:
      self.rect.x = RES[0] - self.size[0]
```

Clean clode

Having all the most regularly used global variables clearly labelled here makes our code later on easier to read. Also, if we want to change the size of something, we only need to do it here and it will work everywhere.

```
class Alien(pygame.sprite.Sprite):
  def __init__(self):
    pygame.sprite.Sprite.__init__(self)
    self.size = (ALIEN_SIZE)
    self.rect = self.image.get_rect()
    self.has_moved = [0, 0]
    self.vector = [1, 1]
    self.travel = [(ALIEN_SIZE[0] - 7), ALIEN_SPACER]
    self.speed = 700
    self.time = pygame.time.get_ticks()
  def update(self):
    if GameState.alien_time - self.time > self.speed:
      if self.has_moved[0] < 12:
        self.rect.x += self.vector[0] * self.travel[0]
        self.has_moved[0] +=1
      else:
        if not self.has_moved[1]:
          self.rect.y += self.vector[1] * self.travel[1]
        self.vector[0] *= -1
        self.has_moved = [0, 0]
        self.speed -= 20
        if self.speed <= 100:
          self.speed = 100
      self.time = GameState.alien_time

class Ammo(pygame.sprite.Sprite):
  def __init__(self, color, (width, height)):
    pygame.sprite.Sprite.__init__(self)
    self.image = pygame.Surface([width, height])
    self.image.fill(color)
    self.rect = self.image.get_rect()
```

Rain bullets

The Ammo class is short and sweet. We only need a few initialising attributes and the update method checks if it's still on the screen. If not, it's destroyed.

The Code CODING PIVADERS

```
    self.speed = 0
    self.vector = 0
  def update(self):
    self.rect.y += self.vector * self.speed
    if self.rect.y < 0 or self.rect.y > RES[1]:
      self.kill()

class Block(pygame.sprite.Sprite):
  def __init__(self, color, (width, height)):
    pygame.sprite.Sprite.__init__(self)
    self.image = pygame.Surface([width, height])
    self.image.fill(color)
    self.rect = self.image.get_rect()
class GameState:
  pass
```

```
class Game(object):
  def __init__(self):
    pygame.init()
    pygame.font.init()
    self.clock = pygame.time.Clock()
    self.game_font = pygame.font.Font(
    'data/Orbitracer.ttf', 28)
    self.intro_font = pygame.font.Font(
    'data/Orbitracer.ttf', 72)
    self.screen = pygame.display.set_mode([RES[0], RES[1]])
    self.time = pygame.time.get_ticks()
    self.refresh_rate = 20
    self.rounds_won = 0
    self.level_up = 50
    self.score = 0
```

Groups

This long list of groups we're creating are essentially sets. Each time we create one of these items, it's added to the set so it can be tested for collisions and drawn with ease.

```
    self.lives = 2
    self.player_group = pygame.sprite.Group()
    self.alien_group = pygame.sprite.Group()
    self.bullet_group = pygame.sprite.Group()
    self.missile_group = pygame.sprite.Group()
    self.barrier_group = pygame.sprite.Group()
    self.all_sprite_list = pygame.sprite.Group()
    self.intro_screen = pygame.image.load(
    'data/start_screen.jpg').convert()
    self.background = pygame.image.load(
    'data/Space-Background.jpg').convert()
    pygame.display.set_caption('Pivaders - ESC to exit')
    pygame.mouse.set_visible(False)
    Player.image = pygame.image.load(
    'data/ship.png').convert()
    Player.image.set_colorkey(BLACK)
    Alien.image = pygame.image.load(
    'data/Spaceship16.png').convert()
    Alien.image.set_colorkey(WHITE)
    GameState.end_game = False
    GameState.start_screen = True
    GameState.vector = 0
    GameState.shoot_bullet = False
  def control(self):
    for event in pygame.event.get():
      if event.type == pygame.QUIT:
        GameState.start_screen = False
        GameState.end_game = True
      if event.type == pygame.KEYDOWN \
      and event.key == pygame.K_ESCAPE:
        if GameState.start_screen:
```

Control

Taking care of keyboard input is the control method. It checks for key events and acts accordingly depending whether we're on the start screen or playing the game.

```
            GameState.start_screen = False
            GameState.end_game = True
            self.kill_all()
        else:
            GameState.start_screen = True
    self.keys = pygame.key.get_pressed()
    if self.keys[pygame.K_LEFT]:
      GameState.vector = -1
    elif self.keys[pygame.K_RIGHT]:
      GameState.vector = 1
    else:
      GameState.vector = 0
    if self.keys[pygame.K_SPACE]:
      if GameState.start_screen:
        GameState.start_screen = False
        self.lives = 2
        self.score = 0
        self.make_player()
        self.make_defenses()
        self.alien_wave(0)
      else:
        GameState.shoot_bullet = True
  def splash_screen(self):
    while GameState.start_screen:
      self.kill_all()
      self.screen.blit(self.intro_screen, [0, 0])
      self.screen.blit(self.intro_font.render(
      "PIVADERS", 1, WHITE), (265, 120))
      self.screen.blit(self.game_font.render(
      "PRESS SPACE TO PLAY", 1, WHITE), (274, 191))
      pygame.display.flip()
```

"Remember, the main loop happens 20 times a second – so the fact we don't call for the screen to update right at the end of the loop is of no consequence"

```
        self.control()
    def make_player(self):
        self.player = Player()
        self.player_group.add(self.player)
        self.all_sprite_list.add(self.player)
    def refresh_screen(self):
        self.all_sprite_list.draw(self.screen)
        self.refresh_scores()
        pygame.display.flip()
        self.screen.blit(self.background, [0, 0])
        self.clock.tick(self.refresh_rate)
    def refresh_scores(self):
        self.screen.blit(self.game_font.render(
        "SCORE " + str(self.score), 1, WHITE), (10, 8))
        self.screen.blit(self.game_font.render(
        "LIVES " + str(self.lives + 1), 1, RED), (355, 575))
    def alien_wave(self, speed):
        for column in range(BARRIER_COLUMN):
            for row in range(BARRIER_ROW):
                alien = Alien()
                alien.rect.y = 65 + (column * (
                ALIEN_SIZE[1] + ALIEN_SPACER))
                alien.rect.x = ALIEN_SPACER + (
                row * (ALIEN_SIZE[0] + ALIEN_SPACER))
                self.alien_group.add(alien)
                self.all_sprite_list.add(alien)
                alien.speed -= speed
    def make_bullet(self):
        if GameState.game_time - self.player.time > self.player.speed:
            bullet = Ammo(BLUE, BULLET_SIZE)
            bullet.vector = -1
```

Refreshing the screen

You need to carefully consider the way in which you update the screen. Blitting the background between actor movements is vital for clean animation.

```
        bullet.speed = 26
        bullet.rect.x = self.player.rect.x + 28
        bullet.rect.y = self.player.rect.y
        self.bullet_group.add(bullet)
        self.all_sprite_list.add(bullet)
        self.player.time = GameState.game_time
      GameState.shoot_bullet = False
  def make_missile(self):
      if len(self.alien_group):
        shoot = random.random()
        if shoot <= 0.05:
          shooter = random.choice([
          alien for alien in self.alien_group])
          missile = Ammo(RED, MISSILE_SIZE)
          missile.vector = 1
          missile.rect.x = shooter.rect.x + 15
          missile.rect.y = shooter.rect.y + 40
          missile.speed = 10
          self.missile_group.add(missile)
          self.all_sprite_list.add(missile)
  def make_barrier(self, columns, rows, spacer):
      for column in range(columns):
        for row in range(rows):
          barrier = Block(WHITE, (BLOCK_SIZE))
          barrier.rect.x = 55 + (200 * spacer) + (row * 10)
          barrier.rect.y = 450 + (column * 10)
          self.barrier_group.add(barrier)
          self.all_sprite_list.add(barrier)
  def make_defenses(self):
      for spacing, spacing in enumerate(xrange(4)):
        self.make_barrier(3, 9, spacing)
```

Guns 'n' ammo

Bullets and missiles use the same parent class. We change a few key attributes originally initialised to create the behaviour we need; eg the vector for each is opposite.

```
  def kill_all(self):
    for items in [self.bullet_group, self.player_group,
    self.alien_group, self.missile_group, self.barrier_group]:
      for i in items:
        i.kill()
  def is_dead(self):
    if self.lives < 0:
      self.screen.blit(self.game_font.render(
      "The war is lost! You scored: " + str(
      self.score), 1, RED), (250, 15))
      self.rounds_won = 0
      self.refresh_screen()
      pygame.time.delay(3000)
      return True
  def win_round(self):
    if len(self.alien_group) < 1:
      self.rounds_won += 1
      self.screen.blit(self.game_font.render(
      "You won round " + str(self.rounds_won) +
      "  but the battle rages on", 1, RED), (200, 15))
      self.refresh_screen()
      pygame.time.delay(3000)
      return True
  def defenses_breached(self):
    for alien in self.alien_group:
      if alien.rect.y > 410:
        self.screen.blit(self.game_font.render(
        "The aliens have breached Earth defenses!",
        1, RED), (180, 15))
        self.refresh_screen()
        pygame.time.delay(3000)
```

Dead or alive

Probably two of the most important questions are answered here – is the player dead or did you win the round?

```
        return True
  def calc_collisions(self):
    pygame.sprite.groupcollide(
    self.missile_group, self.barrier_group, True, True)
    pygame.sprite.groupcollide(
    self.bullet_group, self.barrier_group, True, True)
    if pygame.sprite.groupcollide(
    self.bullet_group, self.alien_group, True, True):
      self.score += 10
    if pygame.sprite.groupcollide(
    self.player_group, self.missile_group, False, True):
      self.lives -= 1
  def next_round(self):
    for actor in [self.missile_group,
    self.barrier_group, self.bullet_group]:
      for i in actor:
        i.kill()
    self.alien_wave(self.level_up)
    self.make_defenses()
    self.level_up += 50
  def main_loop(self):
    while not GameState.end_game:
      while not GameState.start_screen:
        GameState.game_time = pygame.time.get_ticks()
        GameState.alien_time = pygame.time.get_ticks()
        self.control()
        self.make_missile()
        for actor in [self.player_group, self.bullet_group,
        self.alien_group, self.missile_group]:
          for i in actor:
            i.update()
```

Main loop

This is the business end of our application. This loop executes 20 times a second. It needs to be logical and easy for another coder to understand.

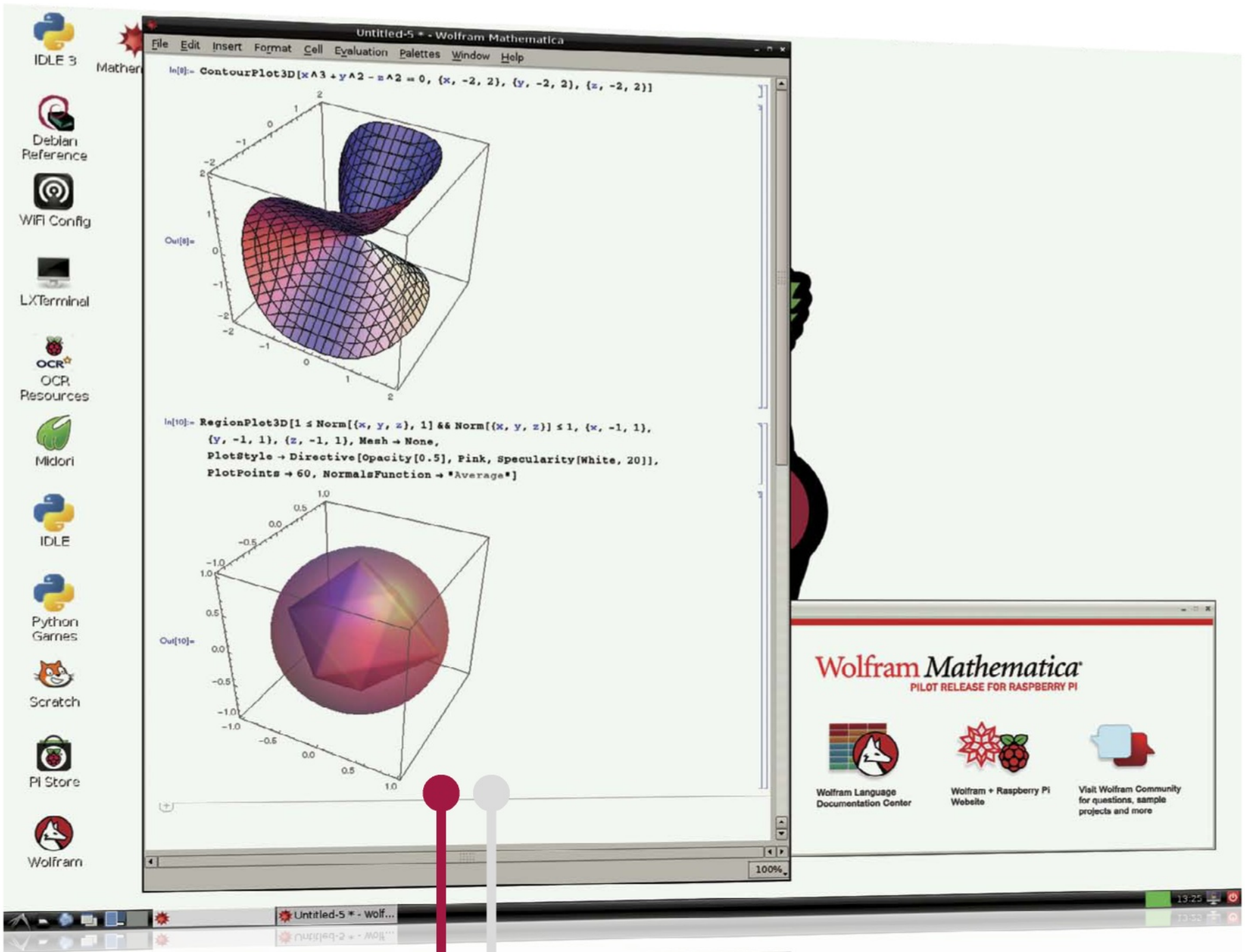
```
            if GameState.shoot_bullet:
              self.make_bullet()
            self.calc_collisions()
            if self.is_dead() or self.defenses_breached():
              GameState.start_screen = True
            if self.win_round():
              self.next_round()
            self.refresh_screen()
          self.splash_screen()
        pygame.quit()

if __name__ == '__main__':
  pv = Game()
  pv.main_loop()
```

Start the game

The very last thing we do is create a Game object and call the main loop. Besides our constants, this is the only code that sits outside a class.

What is Mathematica?

A coding language from Wolfram supplied free to all Raspberry Pis. Why should you care?



"The ultimate platform for doing anything that involves maths or computational modelling"

Q I'm going to be completely honest, I really have no idea what Mathematica is. Not even sure I've heard of Wolfram. So let's start with that. What is Wolfram?

A Wolfram Research is a company that primarily makes Mathematica and was founded by a man called Stephen Wolfram. It also owns Wolfram Alpha, the smart search engine that Siri in the iPhone uses. They're very big on numbers, apparently.

Q Numbers and a search engine, how exactly do they relate to each other?

A Due to the way Wolfram Alpha collects data, it can collect datasets for Mathematica to use in its database.

Q I'm not quite sure what that even means. So here's the big question: what is Mathematica?

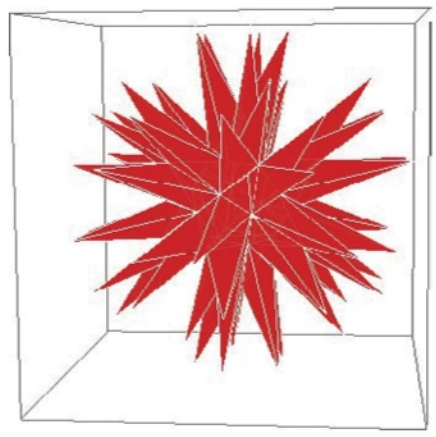
A Mathematica is a very advanced computational program and framework that lets you add very complex mathematics into code and spreadsheets a bit more easily than you would with the standard tools of the language or spreadsheet software. It can be integrated into interactive documents, websites and more. It's billed as the ultimate platform for doing anything that involves maths or computational modelling.

Q So it's a plug-in that makes maths stuff easier?

A Not exactly, as it can be used on its own to perform calculations and computational models. Maths can be a lot trickier than adding and subtracting, with complicated formulas to create 2D and 3D graphs needing some way to be interpreted. It can also handle numbers and variables changing, to redraw models on the fly.

What you can do...

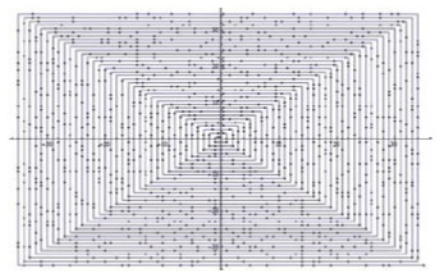
Some excellent Pi Mathematica projects



Echidnahedron

This is a simple shape that resembles the Mathematica logo. Create it with:

```
Graphics3D[{Opacity[.8],
Glow[RGBColor[1,0,0]],
EdgeForm[White],Lighting
->None,PolyhedronData
["Echidnahedron","Faces"]}]
```



Ulam spiral

An Ulam spiral is a way of visualising prime numbers. Create it with:

```
s = {Re@#, Im@#} & /@
 Fold[Join[#1, Last@#1
+ I^#2 Range@#2/2] &, {0},
  Range@140]; ListPlot[s,
Joined -> True,
 Epilog -> {Point /@
s[[Prime@Range@PrimePi@
Length@s]]}]
```

Q Okay, fair enough. Still, why use this rather than some of the maths modules in your spreadsheets or languages?

A While some of that is fairly powerful, Mathematica is purely developed to handle mathematics. It's very optimised and includes a lot of tools and computational functions that don't exist in other maths modules. It's also designed to be relatively easy to work with, allowing you to create complicated models a bit more easily than other languages.

Q Right, so complicated maths made relatively easier. What's the big deal with it being on the Raspberry Pi, though?

A Well for starters, it's free. Very free. Normally it would cost a lot of money to license Mathematica. £200 outright or £95 a year for personal use, and that's the cheapest option. So that's amazing to begin with. Secondly, it's all part of the Raspberry Pi Foundation's commitment to education. Mathematica on the Pi will let kids and adults learn about computer-based mathematics (CBM).

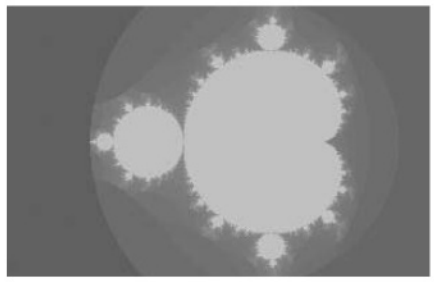
Q Wait, you've lost me again. Computer-based mathematics?

A The concept behind computer-based mathematics is that schools and curriculums are concentrating too much on calculating numbers, not the maths behind it. The CBM movement is about promoting the understanding and applications of maths, while letting computers do the actual computing. This is the same way it's used in the real world. The project is also sponsored by Wolfram.

Q Understanding maths rather than calculating it?

A It goes back to what we said about maths being a lot more complicated than just adding and subtracting.

What you can do...



Fractals

Fractals can make stunning shapes when they are constructed properly:

```
Image@Compile[{},
 Block[{i, x, p},
  Table[i = 0; x = 0. I;
p = r + I c;
   While[Abs@x
   Tanh[Power[i/9^3,
(7)^-1]], {c, -1, 1, .01},
{r, -2, 1,.01}]]][]
```

Understanding the more advanced maths concepts will allow you to use them better in general. In any job where you'd be using these advanced concepts, you'd have computers perform the actual calculations anyway. There's always the age-old question in classrooms about how integrating a formula will help you in the real world, and CBM aims to help explain that along the way.

Q And all this can be achieved just by opening up Mathematica and giving it to kids?

A Well now, CBM also includes an entirely new mathematics curriculum which makes use of Mathematica. Mathematica will be the tool to help people follow along with the courses they've laid out or proposed, and you won't need to spend a penny on actually buying the software for yourself.

Q Do I have to use this very expensive tool just for educational purposes, though?

A Nothing is stopping you from using Mathematica as you see fit on the Raspberry Pi. However, the Pi's hardware itself may be the limiting factor. Also it's licensed on the Pi for non-commercial use only, so you'd be breaking the law if you decided to use it in an enterprise setting. For hobby programmers, though, anything goes.

Q It's free on the Raspberry Pi, though – doesn't that make it free software?

A The traditional definition of free software doesn't apply to Mathematica. Whereas free software is completely free for you to use and modify as you see fit, Mathematica is closed source and licensed under Wolfram's own proprietary licence. It's free as in beer, but not free as in speech.

"The CBM movement is about promoting the understanding and applications of maths, while letting computers do the actual computing"

Q Wait, but it's running on Linux? Isn't all of Linux free and open source?

A Yes. However, the software running on it doesn't have to be free or open either. There's a lot of non-free software, paid or not, that runs on Linux.

Q Doesn't the Raspberry Pi Foundation support Linux and open source, though?

A Yes it certainly does. However, the Raspberry Pi was created with the express purpose of being able to help kids learn computer science, much like the BBC Micros did in the 1980s. Linux being on the Pi is more of a convenient solution than anything else.

Q So if it's just to do with convenience, how are they actually supporting free and open source software?

A In 2013 alone, the Raspberry Pi Foundation spent about £1 million on open source projects. This included a lot of core Linux-based projects, such as the kernel itself.

Q Ah, Okay. Well, it all sounds excellent to me. Where do I sign up? I have the burning desire to model a hypercube.

A You just need to install it with the following terminal command:

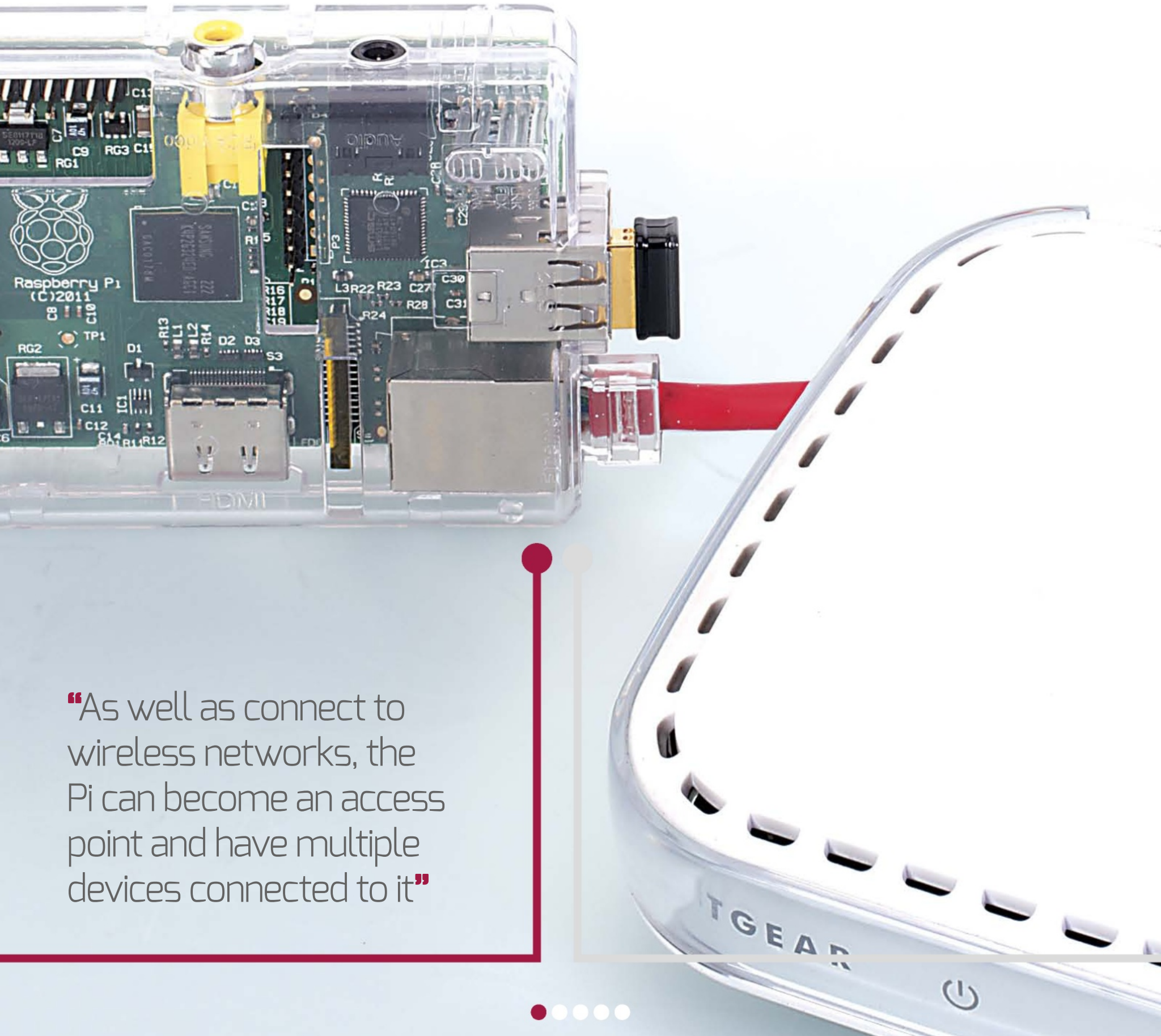
```
$ sudo apt-get update && sudo
apt-get install wolfram-engine
```

It will appear under the Education section of the menu.

"In 2013 alone, the Raspberry Pi Foundation spent about £1 million on open source projects"

Set up a wireless access point with a Raspberry Pi

How to wirelessly connect to your Raspberry Pi, or any existing network connected to it



"As well as connect to wireless networks, the Pi can become an access point and have multiple devices connected to it"

A Raspberry Pi with a wireless dongle that supports Access Point mode is a very versatile tool. As well as connect to existing wireless networks, the Raspberry Pi can become an access point and have multiple devices connected to it. It can either have its own network or bridge onto an existing one and make it wireless. One potential application of this is to create a wireless access point for guests, with appropriate firewalling rules to make sure they can't connect to anything on your LAN. Another is simply connecting to it over wireless rather than Ethernet, which can be useful if the Ethernet port is already in use. For example, a Raspberry Pi acting as a portable PXE server (used to boot a computer over the network) would have an Ethernet cable connected to the computer it's booting, but you might need to log in and see what's going on at the same time.

THE PROJECT ESSENTIALS

Latest Raspbian Image
raspberrypi.org/downloads

A router or switch

A USB wireless dongle that supports Access Point mode

A Wi-Fi device to test with

01 Install the required software

Log into the Raspbian system with the username pi and the password raspberry. Get the latest package lists using the command:

```
sudo apt-get update
```

We'll be using hostapd, which is a daemon that handles access point management and authentication. We'll also need bridge utils, which will be used to bridge the Ethernet and wireless interfaces together. iw is used to get information about the wireless interface, and dnsmasq is used as a DHCP server. Install these with:

```
sudo apt-get install hostapd bridge-utils iw dnsmasq
```

"One potential application of this is to create a wireless access point for guests, with appropriate firewalling"

02 Check for AP mode

It's crucial that you have a wireless dongle that supports Access Point (AP) mode. To verify that, connect the wireless dongle and type iw list. There will be a section titled 'Supported interface modes'. Ensure that AP mode is in that list. Our expert's output looked like this:

```
Supported interface modes:
    * IBSS
    * managed
    * AP
    * AP/VLAN
    * WDS
    * monitor
    * mesh point
```

03 Bridge interfaces

We'll bridge the Ethernet interface and the wireless interface so they can share the same IP address and traffic can pass between them. To do this, edit /etc/network/interfaces (using sudo) and change it to:

```
auto lo br0
iface lo inet loopback
allow-hotplug eth0
iface eth0 inet manual
allow-hotplug wlan0
iface wlan0 inet manual
iface br0 inet dhcp
    bridge_ports eth0 wlan0
    bridge_waitport 0
```

At the moment, we'll be bridging onto an existing network, but we will set up our own later on. Reboot using sudo reboot so that the changes take effect.

Skills to learn

Connect
It's a very useful and convenient form of device connectivity and used by many add-on peripherals.

Networking
The world of networking is something of a dark art. Projects like this make it much less like magic.

"At the moment, we'll be bridging onto an existing network, but we will set up our own later"

04 Configure hostapd

We first need to edit /etc/default/hostapd to start the daemon on boot. Uncomment the 'DAEMON_CONF' option and replace it with:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

We then need to create the hostapd config file at the path specified above. It should have the following contents:

```
interface=wlan0
bridge=br0
driver=nl80211
country_code=UK
ssid=pinet
hw_mode=g
channel=1
wpa=2
wpa_passphrase=super_secret
wpa_key_mgmt=WPA-PSK
ieee80211n=1
wmm_enabled=1
```

Note that if you don't have a wireless N capable device, or are having stability issues, you'll want to leave out the last two lines. You may also need to change the wireless channel if you are having stability issues. Once you've done this, you can start the hostapd daemon with `sudo /etc/init.d/hostapd start`.

05 Connect to the network

The SSID in the config file is the network name, and the WPA Passphrase is the password. When you connect to the network, the connection will be bridged to the existing wired network and you should receive an IP address from your existing router doing DHCP. Congratulations on creating an access point!

"The SSID in the config file is the network name, and the WPA Passphrase is the password"

06 Create an independent network

We're now going to change the network configuration to be independent from your existing network. The first thing to do is to pick a private address range that you'd like to use. For this example, we'll use 192.168.0.0/24. However, we would recommend using a random address range in case you ever want to route to other networks. The first thing we'll do is assign a static address to the bridge interface. Change the bridge section of /etc/network/interfaces to look as follows:

```
iface br0 inet static
    bridge_ports eth0 wlan0
    bridge_waitport 0
    address 192.168.0.1
    network 192.168.0.0
    netmask 255.255.255.0
```

07 Configure a DHCP server

We installed dnsmasq in the first step. Configuring it is really simple. Start by taking a copy of the default config:

```
sudo cp /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

Then edit /etc/dnsmasq.conf to contain the following lines:

```
interface=br0
dhcp-range=192.168.0.2,192.168.0.254,255.255.255.0,12h
```

Reboot for the changes to take effect. You'll want to disconnect the Raspberry Pi from your existing network at this point, otherwise there will be two DHCP servers on the network.

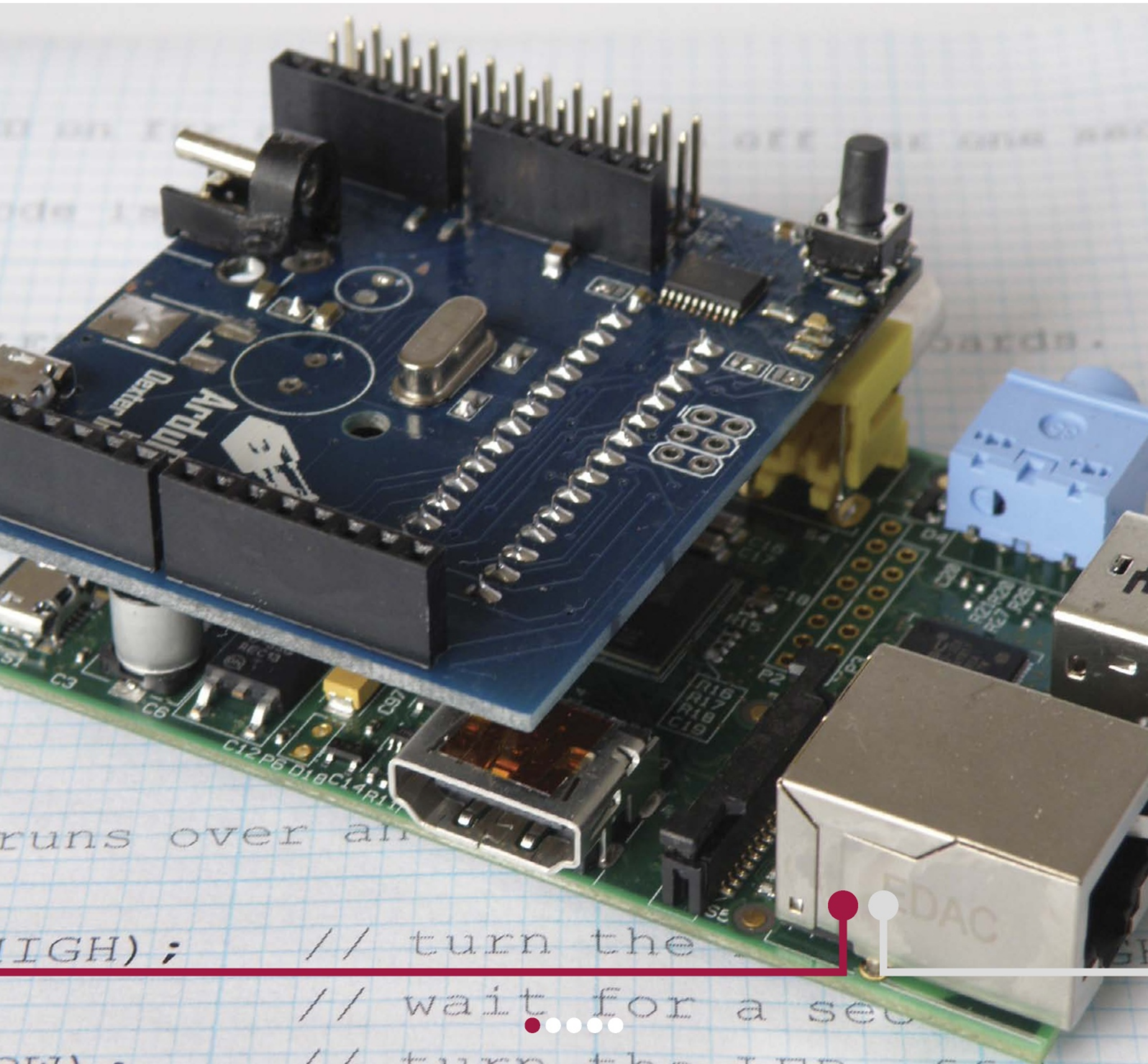
08 Try it out

Devices on the wireless interface will be able to talk to devices on the wired interface, and any device plugged into the Pi will get an IP address so that it can talk to the other devices.

"We would recommend using a random address range in case you ever want to route to other networks"

Arduberry

Anyone can build an add-on board for the Raspberry Pi. Here's the story of Arduberry, a bridge between Raspberry Pi and Arduino...



So where did the idea come about to create a way to attach Arduino shields to the Raspberry Pi?

We had a scratch-your-own-itch issue, where we were trying to prototype some new stuff for the Raspberry Pi. One day we were trying to solve a friend's problem and just attach a GPS to the Raspberry Pi and do some data logging and there weren't any great options for this. When we first did it, it was before the Gertduino came out. So we came up with a quick hack on a breadboard, and we decided we wanted to make it into something everybody could use. It was also around the time, I think, the guy who does WiringPi [Gordon Henderson], he sort of came up with a way you can program the Arduino at the same time, maybe in October or November. He'd done this through the SPI that's on the Raspberry Pi.

So it was fortunate that all these things happened at once. We were a little slow getting out of the gate with Arduberry, though, as we had a couple of projects going on. We just wanted to make something super-simple that required no configuration and would just work if you wanted to put a GPS shield or any other Arduino shield on there you could think of. This would allow you to then really use the Raspberry Pi as a programming computer.

What have you learned since starting Dexter Industries?

You know, actually just about everything. When I was an undergrad, I used Linux for physics projects, but I hadn't used it until we got a Raspberry Pi in late 2012, early 2013.

"We just wanted to make something super-simple that required no configuration and would just work"

John Cole was inspired by the maker movement and started tinkering with electrical engineering concepts in 2009. He now heads up Dexter Industries, Inc

If you like

Similar but different, the Gertduino allows you to connect an Arduino Uno-like board to the Raspberry Pi instead of just connecting various shields.

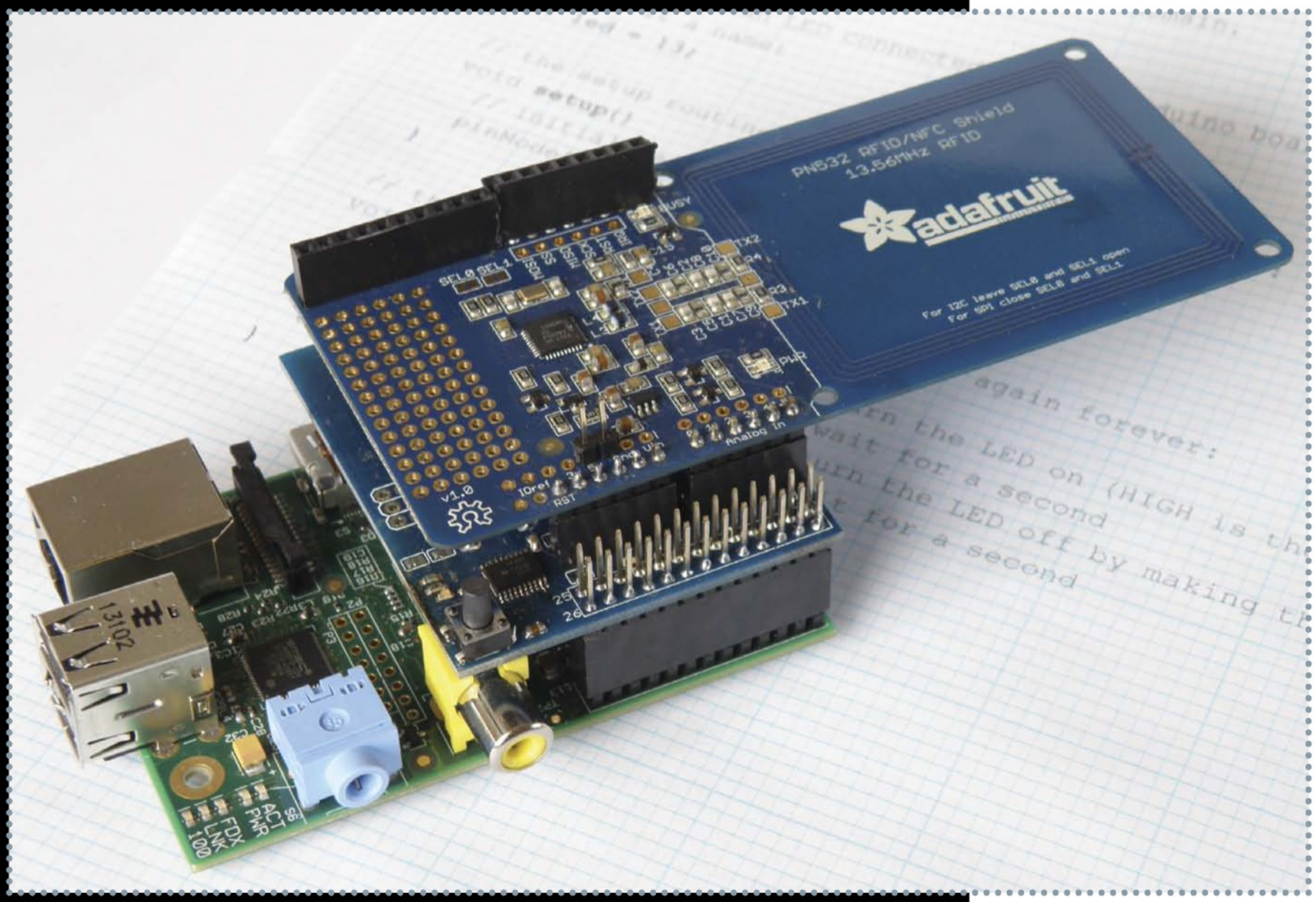
Further reading

To learn more about Arduberry, visit the Dexter Industries site: **bit.ly/1tzgjWd**

And so, that was a really fun experience, just sort of to get back into using a Linux distro. It was also an excuse to learn Python, because I'd never used that before. Before that, I really just picked up circuit design and electronics on the fly, learning as I went. Now I have a couple engineers that work for me and they know a little bit more about what they're doing. We all sort of get a different perspective as we don't have a formal background in electronics or programming; we can look at it maybe from a different perspective.

You Kickstarted the Arduberry. What's the response been to that?

It's been good. We've got something like 500 and some backers as of this morning [Ed: it closed at 672]. So that's great; it was originally going to be a side project and we



Below One or more Arduino shields can be stacked on top of the Arduberry

really wanted to get 100 so that we could have that critical mass to send off and do the tooling and PCBs. So this has been really unexpected, but fantastic. We'll carry this and try and put it out to our distributors.

When we first started the Kickstarter, the most fun part about it is that you get a lot of emails. Especially when things are hot, you get an email every couple of minutes saying someone has backed you and that they're part of your community. It's kind of exciting. What we did was we took an Arduberry and we hooked it up to the Raspberry Pi. The Raspberry Pi scraped the web data and then the Arduberry was hooked up to a car airhorn we had in the lab. So every time we got a new backer, it would play the *Dukes of Hazzard* car horn!

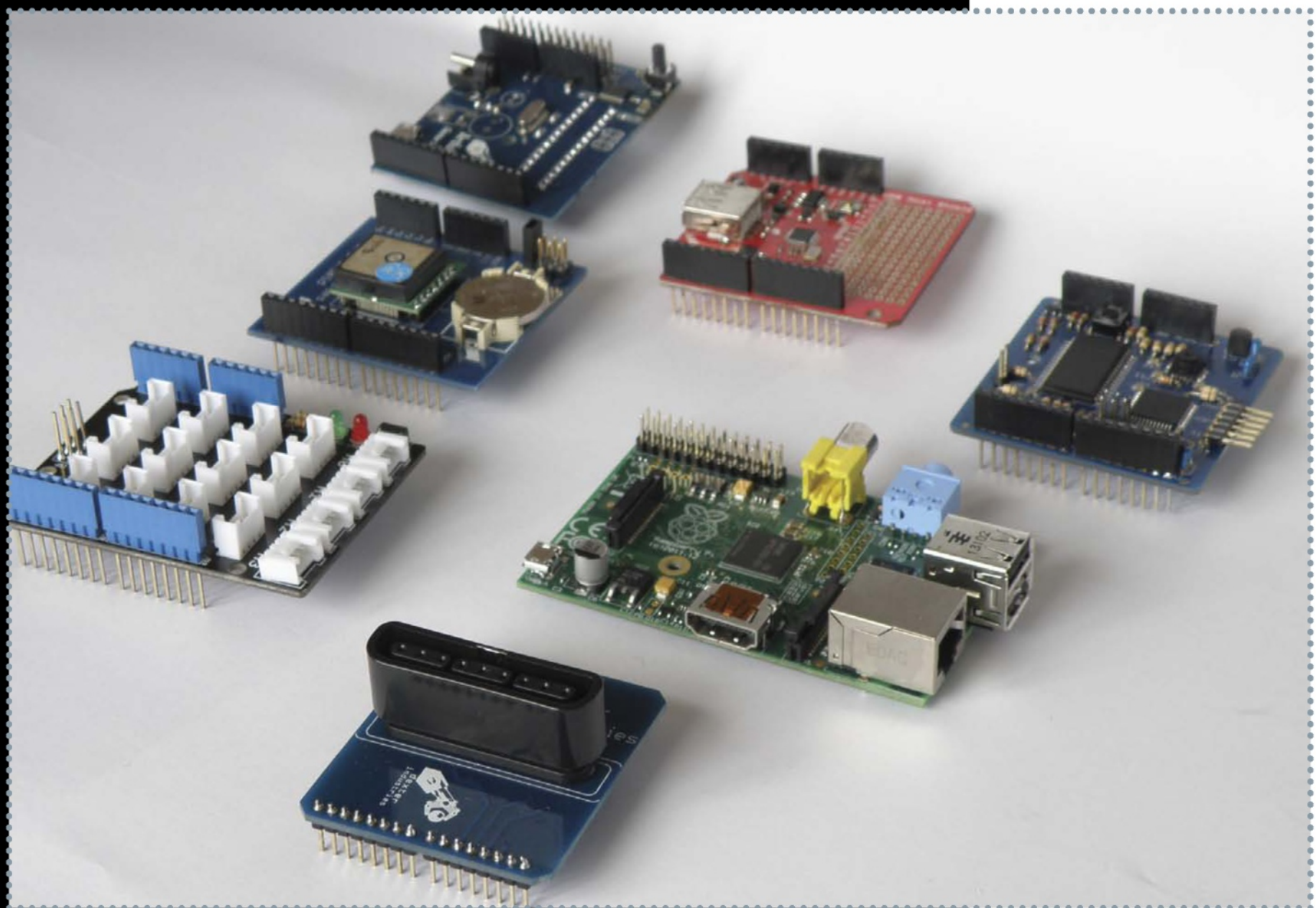
> "The Raspberry Pi scraped the web data and then the Arduberry was hooked up to a car airhorn"

We've seen some impressive photos and videos of you stacking shields on the Pi. How many have you managed?

Everything that we own. We've tried it out with maybe ten different shields and everything seems to work, so all the usual stuff you'll do with an Arduino, from the serial to the I2C to SPI to the digital inputs and that sort of stuff.

After the Arduberry Kickstarter is done, do you have any plans for future projects?

I think after we launch this one, we've got one or two more Raspberry Pi projects coming out, and we may do another Kickstarter for one of them. Six months ago we did a project called BrickPi. It was our first robotics product for the Raspberry Pi and it used LEGO. We had a phenomenal response for that, but one of the limiting factors, I think, is it's sort of uncovered that there's a big demand for using this stuff to teach robotics. And so, we're working on an



all-in-one platform that won't be LEGO-based, but it will still be a simple Raspberry Pi robot.

We're going to call it GoPiGo. There are a lot of people out there that sort of contact us and ask us if it works with anything else other than LEGO, because they like the idea and they don't want to pay an extra $250 for a full LEGO robotics kit just to use it.

"We're working on an all-in-one platform that won't be LEGO-based, but it will still be a simple Raspberry Pi robot"

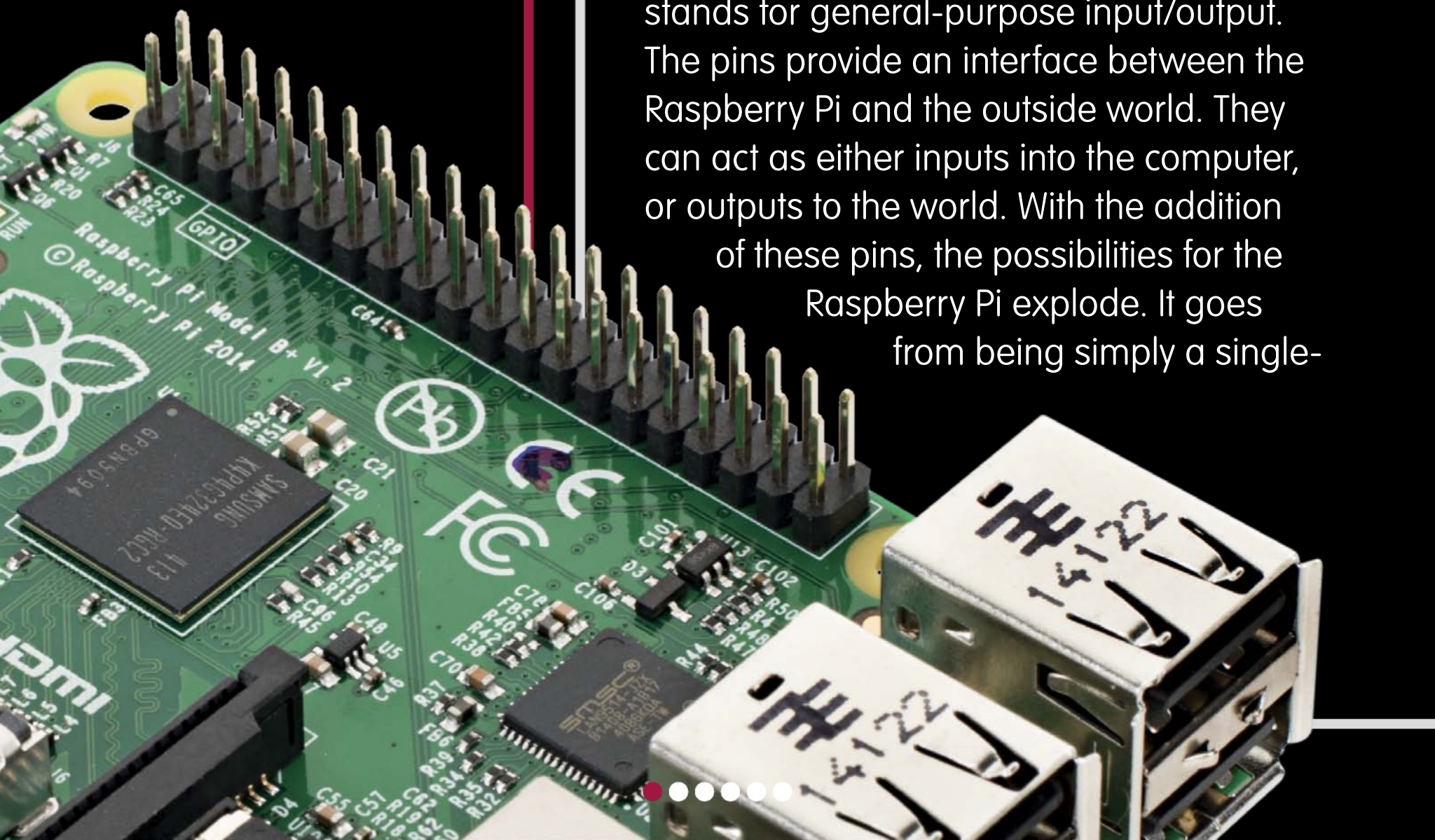
Above The Arduberry is compatible with all manner of Arduino shields, bridging the gap between Pi and Arduino

Talking to the outside world

How to use Python to get your Raspberry Pi to talk to the outside world by using the GPIO pins

"The pins provide an interface between the Raspberry Pi and the outside world. They can act as either inputs or outputs"

Over the last few issues, we've looked at some fundamental concepts in Python, concerning objects, object-oriented programming and how objects are stored in memory. This time, let's take a look at the Raspberry Pi and one of the unique features not usually offered on single-board computers. Of course, I'm speaking of the GPIO pins. GPIO stands for general-purpose input/output. The pins provide an interface between the Raspberry Pi and the outside world. They can act as either inputs into the computer, or outputs to the world. With the addition of these pins, the possibilities for the Raspberry Pi explode. It goes from being simply a single-



board computer to a project platform. Now you can imagine using it for robotics, or as the smarts within some scientific instrument. Many people are using the Raspberry Pi as the core for home automation projects. It really is only limited by your imagination. We will focus on how to talk to the GPIO pins, rather than how to wire up the electronics. While the Linux kernel has modules to support communicating with the GPIO pins, this isn't enough. You will also need support within whichever programming language you are using. Since this column is about Python programming, we will focus only on those Python modules available for your use. Luckily, if you are using a standard distribution, like Raspbian, then you will have this available to you out of the box. If you are using some other distribution, or if you are really adventurous and have rolled one yourself, you will need to go out and grab a relevant Python module from the web. In this article, we will be looking at the RPi.GPIO module. But this is not your only option.

Before we move on to the actual code, we should take a little time to look at the electronics first. While there is not enough time to do a crash course, there are a few things that everyone should know about how the GPIO pins are implemented on the Raspberry Pi. The first thing to note is that there are two versions of the Raspberry Pi, and the layout of the pins is different on both. You should verify which version you have before actually attaching any wires. There are 26 pins, laid out as two rows of 13. You have a series of ground pins, +5 volts and +3.3 volts power pins, along with all of the actual data pins. There are eight GPIO pins that are available, but they are unprotected. They are designed to use +3.3 volt voltage levels, and anything higher (like +5 volts voltage levels) may damage your Raspberry Pi. The current

"There are eight GPIO pins that are available, but they are unprotected. They are designed to use +3.3 volt voltage levels"

can be configured up to 16mA. The intention is that you would not connect directly to the GPIO pins, but attach an interface board that would handle buffering of inputs and voltage control in order to protect your Raspberry Pi. These pins can be used in multiple different function modes, including I2C, SPI and PWM.

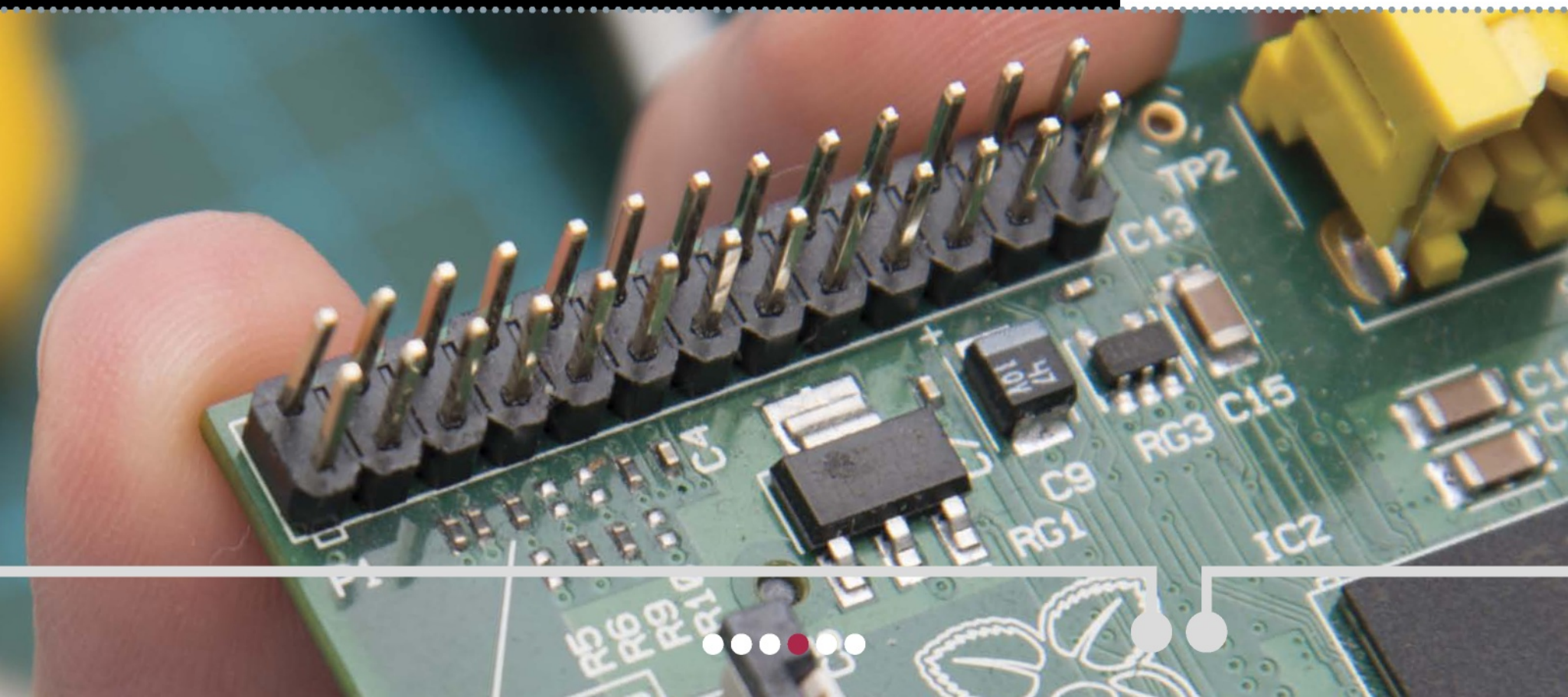
Once you have your circuit all wired up, it is time to start actually talking to it. When you are just starting out, you may want to simply start up the Python interpreter and enter your code directly. That way, you can experiment and try things out. Once you get a feeling for how things work, you will be more comfortable writing Python scripts to do more complicated tasks. The first step is to import the `RPi.GPIO` module. You probably won't want to write this out over and over again, so you will probably want to import it with a line like `import RPi.GPIO as GPIO`. Once imported, you can find out what version of board and RPi you have. The board revision is given by `GPIO.RPI_REVISION`, and the version of the RPi module is given by `GPIO.VERSION`. The next step is to set the numbering system you wish to use to address the pins. You can choose either BOARD (which uses the same numbering scheme as the physical pin numbers) or BCM (which uses the channel numbers on the Broadcom SoC). In most cases, you will want to use the BOARD numbering system, which you can set with the line `GPIO.setmode(GPIO.BOARD)`. The next step is to initialise each pin you wish to use. You need to tell Python how you want to use each pin: whether it is supposed to be an input pin or an output pin. For this article, you will want to set some pins as outputs. Say you want to use pin 8. You can do this with the line `GPIO.setup(8, GPIO.OUT)`. You can even set an initial value for output. If you wanted to start by having the

> "The intention is that you would not connect directly to the GPIO pins, but attach an interface board that would handle buffering of inputs and voltage control"

light turned on, you would use `GPIO.setup(8, GPIO.OUT, initial=GPIO.HIGH)`. You can now start sending data to the pin. You can set the voltage either high or low with `GPIO.output (8, GPIO.HIGH)` or `GPIO.output(8, GPIO.LOW)`. When you start writing more complicated code, you may need to check what state a particular pin is in. You can do this with the command `GPIO.gpio_function(pin)`. It will return one of the following values: GPIO.INPUT, GPIO.OUTPUT, GPIO.SERIAL GPIO.SPI, GPIO.I2C, GPIO.PWM or GPIO.UNKNOWN. Once you are all done, you need to clean up after yourself. You can do this with the command `GPIO.cleanup()`. If you only want to clean up particular channels, you can do so with, say, `GPIO.cleanup(8)`.

Now that you know how to talk to the outside world, next time we will look at how to read from the outside world. Combining this with some matplotlib code will give you the tools to build some basic instrumentation. Until then, play around with more complicated programs to take control of the world around you. Just always be careful when playing with power around your board. You don't want to accidentally cook it and release the magic blue smoke (the electronics nerds among you will get that reference). And always remember to have fun.

"You can choose either BOARD (which uses the same numbering scheme as the physical pin numbers) or BCM (which uses the channel numbers on the Broadcom SoC)"



Below These 26 pins are the key to enabling your Raspberry Pi to talk to the outside world

The Code USING THE GPIO PINS

```
# We will be sending signals out from our Raspberry Pi

# We will need the time module
import time

# First, import the GPIO module
import RPi.GPIO as GPIO

# Set the numbering mode for the pins
GPIO.setmode(GPIO.BOARD)

# Set pin 12 in output mode
GPIO.setup(12, GPIO.OUT)

# Now, turn the LED on, sleep for 1 second, then turn it off
GPIO.output(12, GPIO.HIGH)
time.sleep(1)
GPIO.output(12, GPIO.LOW)

# Now that we are done, we can cleanup
GPIO.cleanup()

##################################################
# We can use PWM to blink our LED

# Set the mode
GPIO.setmode(GPIO.BOARD)

# Set the pin
GPIO.setup(12, GPIO.OUT)
```

setmode

When you first initialise GPIO with the setmode method, you need to select either BOARD or BCM. It's confusing, so stick with the convention that makes sense to you.

The Code USING THE GPIO PINS

```
# Create a PWM instance
p = GPIO.PWM(12, 20) # channel=12, frequency=20Hz

# Start the PWM signal
p.start(1)

# We will let this go until someone presses return
raw_input('Press return to stop:')

# Now we can stop the PWM object and cleanup
p.stop()
GPIO.cleanup()
```

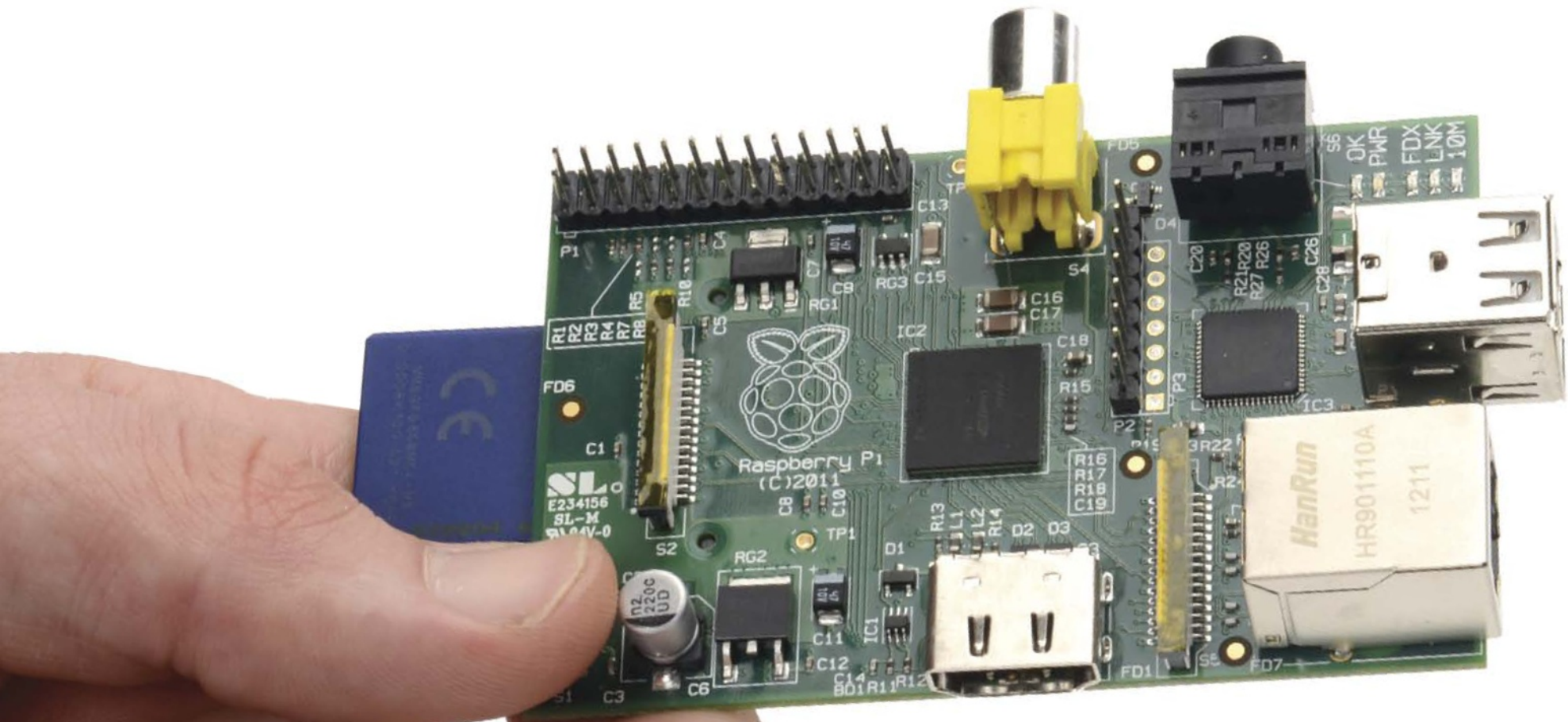
PWM

Pulse Width Modulation is a very powerful tool for controlling everything from LEDs to motors and servos.

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk



The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

Can I connect to my Raspberry Pi from another computer?
Trent via Facebook

You absolutely can; there are a few ways to do so but the quicket and easiest way is to do it via SSH. If you're running Linux or OS X you can use SSH in the terminal, and in Windows you can use software calld Putty.

The first thing you need to do is find out what IP address your Raspberry Pi has. You can find that out by typing ifconfig into the command line. Once you know it, go to the terminal on another computer (or Putty) and enter:

```
$ ssh pi@[IP address]
```

It will ask for a password, which is raspberry, and then you're in!

Is NOOBS the best way to install an operating system on my Raspberry Pi?
Jamie C. via email

It's largely the easiest way to install an operating system. It can take a bit more space on the SD card than performing a more traditional install using the dd command, however you can at least dual boot to a degree and try out other distros very easily.

For the dd method, download the img from the Raspberry Pi website, stick your SD card into a card reader slot and type the following into the command line:

```
$ dd bs=1M if=[location of]/[image].img of=/dev/[location of SD card]
```

Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

Win a FUZE

A modern-day mash-up of the Raspberry Pi and the classic BBC Micro, the FUZE is a programmable computer and electronics workstation designed to teach users how to program hardware and software. There's space just above the keyboard for breakout boards to be added to the FUZE IO board, which in turn makes it easier for you to work with the Pi's GPIO ports. The FUZE comes with FUZE BASIC, Python and Scratch pre-installed, plus some PDF project cards to get you started with them.

What's the best way to install new software to the Raspberry Pi?
Sam via email

There are a few ways to install software on the Pi, the main way on Raspbian is by using the apt-get command in the terminal; such as sudo apt-get install vlc for the VLC media player. Otherwise, there's the Raspberry Pi Store, which is accessible from the desktop – it's an online storefront with free and paid software that will install to your Pi.

Your best bet is to search online for 'install software raspbian' to get details on what package you'd need from apt-get.

Can I play games on the Raspberry Pi?
Jakub via Facebook

There are some games that exist on the Raspberry Pi, and you can find them on the online Raspberry Pi Store to install and play. There's not a huge amount of games on there though, and fewer still that you may have heard of. There are no plans for Steam to be released on the Raspbian version of Debian, unfortunately, and there may never be any plans for it to be released, either.

It's mainly because of the lack of power the Raspberry Pi has. However, Minecraft is available in some form on the Raspberry Pi, which may open the door to more.

We're giving away five FUZEs – for a chance to win one you just have to download FUZE BASIC onto your Pi, have some fun experimenting with it and then email us your most creative FUZE BASIC project (there are more details via the competition link below). To learn more about FUZE BASIC, check out this month's issue of Linux User & Developer.

>> 1st Prize:
FUZE T2-R (with robotic arm, worth £229.99)

>> 2nd Prize:
FUZE T2-A (worth £179.99)

>> Runners-up:
x3 FUZE T2-C (worth £89.99)

Closing date for entries
5 December 2014

More info:
www.linuxuser.co.uk/news/win-a-fuze

Next issue

Get inspired · Expert advice · Easy-to-follow guides

Drive a Bigtrak with your Pi

Plus Pivaders animation and sound, tilt control and more

Get this issue's source code at:
www.linuxuser.co.uk/raspicode

FOR THE GNU GENERATION

www.linuxuser.co.uk

FREE RESOURCE DOWNLOAD EVERY ISSUE •

LinuxUser & Developer

Available from all good newsagents & supermarkets today

ON SALE NOW:

» IBM and Linux » Why you need Python 3 » 4 Great competitions

BUY YOUR ISSUE TODAY

Print edition available at www.imagineshop.co.uk

Digital edition available at www.greatdigitalmags.com

Available on the following platforms

Print | Tablet | Phone | Desktop

facebook.com/LinuxUserUK twitter.com/LinuxUserMag